

**Determination of Design and Operation Parameters
for Upper Atmospheric Research Instrumentation
to Yield Optimum Resolution with Deconvolution**

NASA Grant NAG 1-804

FINAL REPORT

APPENDIX 7

**Dr. George E. Ioup, Principal Investigator
Dr. Juliette W. Ioup, Principal Investigator
Department of Physics
University of New Orleans
New Orleans, LA 70148**



University of New Orleans

Lakefront • New Orleans • Louisiana 70148 • (504) 286-6341

DEPARTMENT OF PHYSICS

Don Tomlin and Stan Carroll

We are enclosing everything for the Engineering Notebook except the Theory and Concept Definitions Section. We hope to express mail that to you on Monday. The only new section we are sending is the Introduction. Everything else is either an addition to sections in the document that was mailed to you on 24 Dec 1991 or changed pages for that document. We have put the date of 31 Jan 1992 on this version. Each addition or insert is marked with a yellow stick-on. Please go through carefully and insert or substitute the pages at the appropriate place in the document. If there is any doubt about any of the insertions or substitutions, please call us. We hope they are obvious enough. The insertions or additions are for the following parts of the document:

- Title Page
- Table of Contents
- Introduction
- Program Description
- Test Plan
- Appendix 1
- Appendix 2 Table of Contents
- Appendix 2C
- Appendix 2D
- Appendix 2E
- Appendix 2F beginning
- Appendix 2F end

31 Jan 1992

UNIVERSITY OF NEW ORLEANS

TRANSFORM DOMAIN SKIPROPE OBSERVER

ENGINEERING NOTEBOOK

**STEVE M. RODRIGUE, ABOLFAZL M. AMINI,
GEORGE E. IOUP, AND JULIETTE W. IOUP**

**DEPARTMENT OF PHYSICS
UNIVERSITY OF NEW ORLEANS
NEW ORLEANS, LA 70148**

31 JAN 1992

Transform Domain Skiprope Observer

University of New Orleans 31 Jan 1992

Engineering Notebook

Table of Contents

Introduction

Theory and Concept Definitions

Algorithm Outline

Flow Chart

Program Description

Main Program

Definition of Variables/Flags

Input/Output

Code Discussion

Subroutine WORK

Definition of Variables/Flags

Input/Output

Code Discussion

Subroutine HANN

Definition of Variables/Flags

Input/Output

Code Discussion

Subroutine MEAN

Definition of Variables/Flags

Input/Output

Code Discussion

Subroutine FOUR1

Definition of Variables/Flags

Input/Output

Code Discussion

Subroutine LSCF

Definition of Variables/Flags

Input/Output

Code Discussion

Subroutine FBAND

Definition of Variables/Flags

Input/Output

Code Discussion

Subroutine READINDATA

Definition of Variables/Flags

Input/Output

Code Discussion

Subroutine ODTF

Definition of Variables/Flags

Input/Output

Code Discussion

Quick Reference

Required External Files

Operator Inputs

Warnings/Abort Situations

Program Development History

Test Plan

Appendices

1. Program Code

2. Test Code and Results

Introduction

Because of the interesting science which can be performed using a satellite attached by a very long tether to a mother vehicle in orbit, such as the Space Shuttle, NASA will deploy TSS-1 (Tethered Satellite System) in 1992. A very long tether (20 km in this case) has the possibility of undergoing oscillations of several different types, or modes, and higher harmonics of these modes. The purpose of this document is to describe a method for detecting the amplitude, frequency, and phase (and predicting future motion in the steady state) of these modes, in particular, the skiprope mode, using tethered satellite dynamics measurements. Specifically the rotation rate data about two orthogonal axes, calculated from output from satellite gyroscopes, are used. The data of interest are the satellite pitch and roll rate measurements.

Aside from understanding tether dynamics, one reason it is important to diagnose and predict the skiprope motion of the tether is related to satellite retrieval. The retrieval mechanism has a limited acceptance angle and the tether skiprope motion can cause angular excursions of the satellite beyond this angle. Several methods are available to damp the skiprope, but for these to be successful, it is necessary that the skiprope amplitude, frequency, and phase be known accurately enough and that future values of the parameters be predicted for the time of application of the corrective procedure.

NASA has determined to use two methods to diagnose skiprope

properties and predict future values. One of these, a Fourier transform domain approach, is the subject of this notebook. The other is a time-domain, state-space method being developed by Martin-marietta. It is described elsewhere. Much of the development and testing of the Fourier algorithm and code has been done by the authors at the University of New Orleans. It is very important, however, to give full credit to the other contributors to the development effort. First and foremost is Mr. Stan Carroll of NASA Marshall Space Flight Center, whose help with this project was essential. Mr. Don Tomlin, Mr. Keith Mowery, Mr. Zack Galaboff, and others from MSFC have also contributed. From the University of Southern Mississippi we have received help from Dr. Grayson Rayborn and Dr. Sam Howard.

The method which is described in this notebook can be modified to diagnose tether skiprope motion when the tethered satellite is spinning. This modification has been accomplished, but it is not discussed in this document. An addendum will be issued to describe it. There is also a version of the current code which contains a subroutine which gives the time history of the skiprope in the data observation window. That modification is also not described in this notebook.

Included in this document are a section on Theory and Concept Definitions, in which some of the background theory and definitions for the method are discussed with references given for further reading. An algorithm outline follows, listing all the general steps of the program and including an overview flow

chart. For the main program and every subroutine, there are definitions of variables/flags, descriptions of the input and output, and discussions of the code. The principal subroutine is WORK, within which the Fourier transform and related calculations are performed. Other subroutines which are either called by WORK or the main program are HANN, which applies the HANN window; MEAN, for mean removal; FOUR1, for the calculation of the fast Fourier transform; LSCF, which performs a least squares curve fit; FBAND, which defines the search band in the transform domain to find the maximum of the transform magnitude; and READINDATA and ODTF for input/output. Also included in the section on program description is a quick reference which gives required external files, operator inputs, and a listing of warnings/abort situations. Following the program description is a section which gives program development history. This section is written both to show the trials which led to the selected algorithms and to inform the reader about other methods which did not perform as well as those adopted. Finally, in the main body of this document is a complete description of the test plan which was executed at the University of New Orleans.

There are two Appendices. The first lists the program code which accomplishes skiprope observation for slowly varying skiprope parameters and prediction of future parameter values for steady state tether motion. Appendix 2, which contains the test code and results, forms the bulk of this document. It is so long that it has its own table of contents. It is broken up into six

parts, Appendices 2.A through 2.F. Appendix 2.A contains the programs for generating model signals. In Appendix 2.B are the results of the ECR verification table. Appendix 2.C lists the programs for ECR testing, while Appendix 2.D lists the programs for systematic testing. Appendix 2.E gives the results of simulation tests, and Appendix 2.F gives the systematic test results at Station 2 and Station 1.

Frequency Domain Skiprope Observer

Algorithm Outline

1) Access the data buffers produced by the preprocessor containing the pitch and roll gyro data.

2) Select a time window of data to be analyzed.

For each of the axes perform the following:

3) Apply Hann window to the data.

4) Calculate the mean.

5) Remove the mean from the original data.

6) Apply Hann window to the mean-removed data to reduce sidelobes in the Fourier transform.

7) Use the FFT to calculate the real and imaginary parts of the Fourier transform of the Hann-windowed data.

8) Calculate the amplitude spectrum of the transform.

Frequency Domain Skiprope Observer

Algorithm Outline - continued

- 9) Search the predefined frequency band to find the maximum in the amplitude spectrum.
- 10) Fit a quadratic to the seven points centered around and including this maximum.
- 11) Use this quadratic to define more accurately the largest value in the amplitude spectrum and its corresponding frequency.
- 12) Calculate the satellite motion amplitude from this maximum amplitude value by a simple conversion. The frequency is the frequency of that motion.
- 13) Use quadratic interpolation to find the real and imaginary values of the Fourier transform at the above calculated frequency.
- 14) Use real and imaginary parts to define the phase of the satellite motion based on the model $\cos(2\pi f t + \phi)$.
- 15) Calculate the time index for the first maximum of the gyro signal ω within the data window.

Algorithm Outline - continued

CALCULATIONS UPON RETURN FROM WORK SUBROUTINE

16) Average the x and y frequencies - call FAVG.

17) Find the reciprocal of FAVG (average period), call TAVG.

18) Calculate constant $WK = TETHER\ LENGTH * TAVG / (360 * PI)$

19) Calculate polarity:

a) Compare time indices of maximum.

b) For x time greater than y time. If time difference is less than 1/2 period, polarity is positive, else polarity is negative.

c) For y time greater than x time. If time difference is less than 1/2 period, polarity is negative, else polarity is positive.

20) For both x and y axes, calculate omega values:

$W (X\ or\ Y) = AMP (Y\ or\ X) * COS(2.0 * PI * FAVG * DT * (I - 1) + PHASE(X\ or\ Y))$

21) For both x and y axes, calculate motion amplitude values

$U (or\ V) = - POLARITY * WK * W (X\ or\ Y)$

22) Calculate time values $TIME = T0 + (START\ INDEX + I - 2) * DT$

YAW MANEUVER CALCULATIONS

24) Time of maximum x value: $TXMAX = DT * (INDEX\ OF\ MAX - 1)$

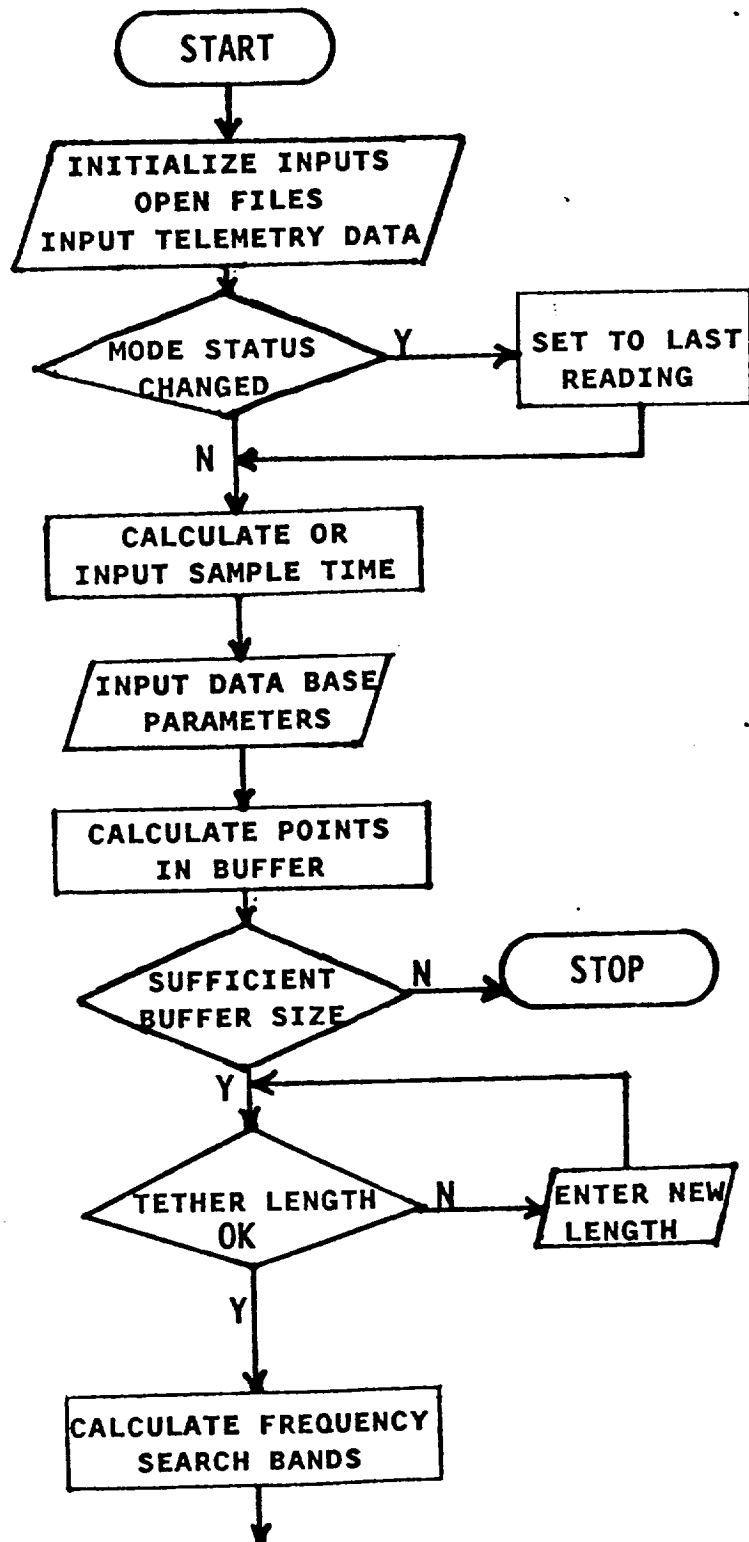
25) Time of maximum y value: $TYMAX = TXMAX + 0.25 * TAVG * TSHIFT + T0$

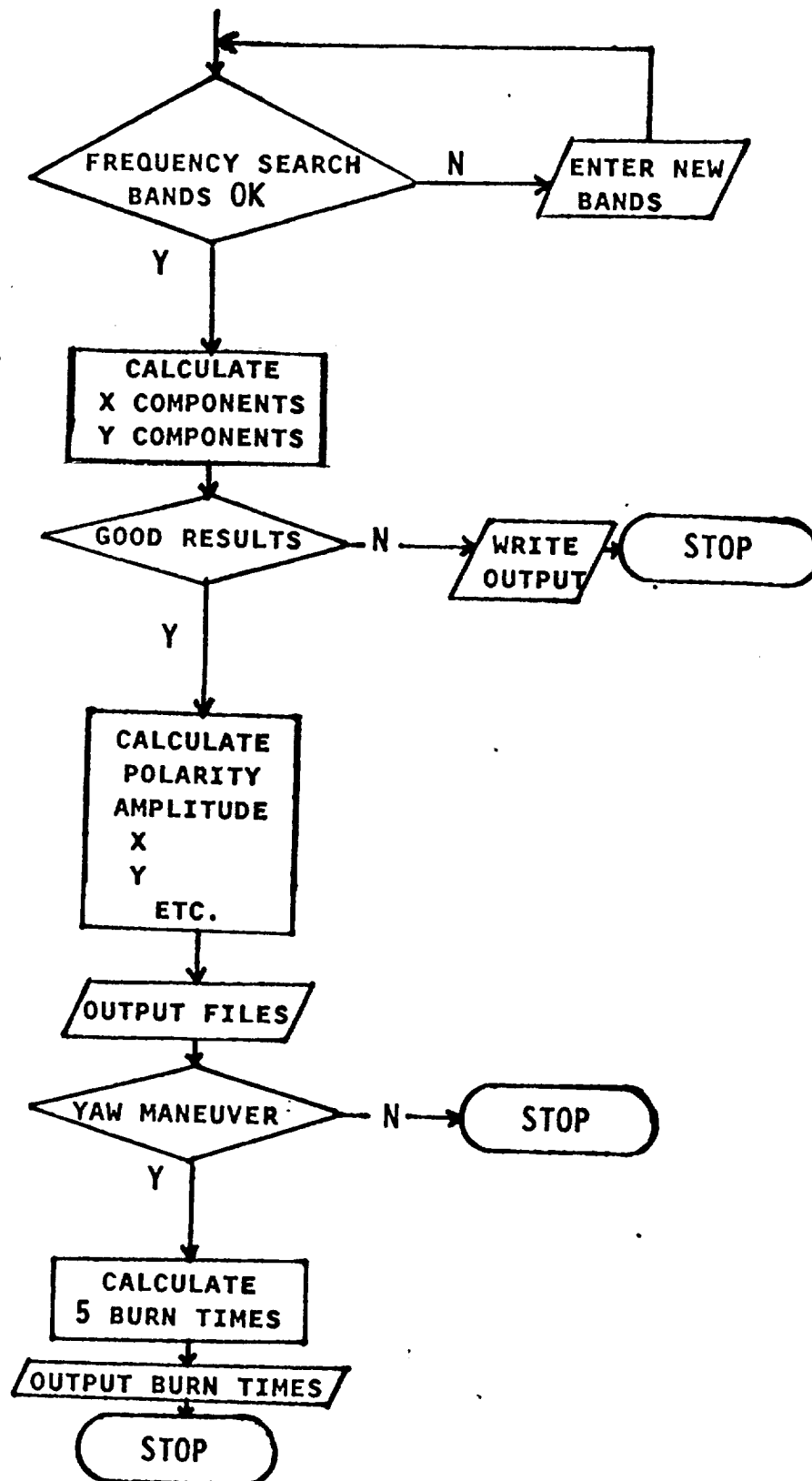
(Note: TXMAX is a relative time, TYMAX is a mission elapsed time.)

26) Time for orbiter maneuver (burn time): $TT = TYMAX + TAVG * (INTEGER)$

(Burn times are integral numbers of periods from TYMAX.)

FREQUENCY DOMAIN SKIPROPE OBSERVER - UNOMSC.FOR
NASA MARSHALL AND THE UNIVERSITY OF NEW ORLEANS





PROGRAM DESCRIPTION

Main Program

Definition of Variables/Flags

Parameters

NDIM - dimension of arrays X, Y, and TLG (set to 3000)
NFT - number of points in the Fourier transform (set to 8192)
NPNT - number of points used in the least squares curve fitting polynomial (set to 7)

Constants

PI - value of pi (set to 3.1415926)
DFR - conversion factor from radians to degrees (set to 57.2957795)
RFD - conversion factor from degrees to radians (set to 0.017453292)

Character Variables

REPLY - character variable of length 1 (replies by the operator to prompts, either y, Y, n, or N)
POLARITY - character variable of length 8 (either positive or negative)

Variables read in from external file PARAM.DAT

PF_SRCH_BAND - percentage used in calculating the frequency search band)
R_ARM - distance from orbiter C.M. to the center line of the deployer boom
ODF_TIME - logical flag to control printing of the time data to output file F_TXYUV.DAT (initially set to .FALSE., i.e., don't print)
ODF_FFT - logical flag to control printing of frequency domain data (FFT's) to the output files F_FFTX.DAT and F_FFTY.DAT (initially set to .TRUE., i.e., print)
DENSITY - tether density in kg/km (set to 8.35 kg/km)
TOTALL - total tether length (set to 22.0 kilometers)
MSAT - satellite mass in kg (set to 510.0 kg)
MORB - orbiter mass in kg (set to 100,000.0 kg)
ALTKM - orbit altitude in km (set to 325.0 km)

Variables read from telemetry preprocessor (or external file IDFTXY.DAT)

T0 - time tag for first point in buffer
TF - time tag for last point in buffer

X(NDIM) - x axis gyro data array (deg/sec)
 Y(NDIM) - y axis gyro data array (deg/sec)
 TLG(NDIM) - tether length array (kilometers)
 JMODE - integer variable, signifying the amcsmode for
 the last time point 'TF'
 amcsmode = 0 indicates no valid data
 amcsmode = 1 indicates passive case
 amcsmode = 2 indicates yaw hold
 amcsmode = 3 indicates spin case
 LF - number of time points stored in each array
 M_FLAG - logical flag to denote if the amcsmode
 changed from 1 or 2 to a 0 or 3 between the
 times of T0 and TF

Other Variables

AMCSMODE - integer variable set to the value of JMODE
 LB - starting time index (requested of operator)
 LE - last time index (requested of operator)
 LEB - total number of data points processed (LEB = LE -
 LB + 1)
 DT - average sample time = $(TF - T0) / (LF - 1)$
 DF - frequency sampling = $1.0 / (NFT * DT)$
 FLOW - low frequency of the frequency search band
 FHIGH - high frequency of the frequency search band
 LEAST - estimate of the number of data points to use
 for a minimum of 3 cycles of skiprope
 LEAST = $INT(6.0 / (FLOW + FHIGH))$
 TLNGTH - tether length used in calculations, first
 estimated from $0.5 * (TLG(1) + TLG(LF))$, and
 finalized as $0.5 * (TLG(LB) + TLG(LE))$
 TSHIFT - time shift from first point in buffer, i.e.,
 T0, and the first time point used in the run
 TSHIFT = $DT * (LB - 1)$
 TMIDPT - time point of the middle of the data window
 TMIDPT = $T0 + DT * ((LB + LE - 2) / 2)$
 S_FLAG - logical flag to control yaw maneuver calcula-
 tions (only set to .TRUE. by direct operator
 reply of 'y' or 'Y' after prompt)
 PSIGN - numerical sign of the polarity (calculated)
 PSIGN = 1.0 indicates positive polarity
 PSIGN = -1.0 indicates negative polarity
 PSIGN = 0.0 indicates inability to predict the
 skiprope frequency
 FREQX - calculated value of the skiprope frequency from
 the x axis gyro data
 AMPX - calculated value of the x gyro rate at FREQX
 PHASEX - phase of FREQX relative to LB
 IWXMAX - time index of the maximum x gyro rate value
 AVGX - mean value of Hann-windowed x axis gyro rate data
 G_FLAGX - logical flag indicating whether the x axis
 values are good, i.e., G_FLAGX is set to
 .TRUE. if FREQX is within the specified fre-

quency search band

FREQY - calculated value of the skiprope frequency from the y axis gyro data

AMPY - calculated value of the y gyro rate at FREQY

PHASEY - phase of FREQY relative to LB

IWYMAX - time index of the maximum y gyro rate value

AVGY - mean value of Hann-windowed y axis gyro rate data

G_FLAGY - logical flag indicating whether the y axis values are good, i.e., G_FLAGY is set to .TRUE. if FREQY is within the specified frequency search band

FAVG - average skiprope frequency = $0.5 * (FREQX + FREQY)$

TAVG - period of the average frequency = $1.0 / \text{FAVG}$

WK - conversion factor from gyro rate values to amplitudes
 $WK = 1000.0 * TLNGTH * TAVG / (360 * PI)$

UMAX - maximum in plane skiprope amplitude
 $UMAX = WK * AMPY$

VMAX - maximum out of plane skiprope amplitude
 $VMAX = WK * AMPX$

TEST - time required to move from the x axis to the y axis (in sec)
 If IWYMAX .GT. IWXMAX, TEST = $DT * (IWYMAX - IWXMAX)$
 If IWXMAX .GT. IWYMAX, TEST = $DT * (IWXMAX - IWYMAX)$

TWX - time domain skiprope signal for the x axis (without proper amplitude scaling)
 $TWX = \cos(2.0 * PI * \text{FAVG} * DT * (I-1) + \text{PHASEX})$

TWY - time domain skiprope signal for the y axis (without proper amplitude scaling)
 $TWY = \cos(2.0 * PI * \text{FAVG} * DT * (I-1) + \text{PHASEY})$

WX - time domain skiprope signal for the x axis (with proper amplitude scaling) $WX = TWX * AMPX$

WY - time domain skiprope signal for the y axis (with proper amplitude scaling) $WY = TWY * AMPY$

U - in plane skiprope amplitude (in meters)
 $U = -PSIGN * WK * AMPY * TWX$

V - out of plane skiprope amplitude (in meters)
 $V = -PSIGN * WK * AMPX * TWY$

T - time tag = $T0 + (LB + I - 2) * DT$

RNROT - number of rotations the orbiter should execute
 $RNROT = (\text{AMIN1}(UMAX, VMAX)) / (2.0 * R_ARM)$

TXMAX - time of maximum x axis gyro rate (should correspond to when the tether is over the orbiter nose) = $DT * (IWXMAX - 1)$

TYMAX - time of maximum y axis gyro rate (should correspond to when the tether is over an orbiter wing) = $TXMAX + 0.25 * TAVG + TSHIFT + T0$

TT - predicted times to execute the yaw maneuver
 $TT = TYMAX + (K-1) * TAVG$ (K is an integer that runs from 1 to 5)

OYAWANG - orbiter yaw axis angle in degrees

BTDEL - burn time delay = $PSIGN * OYAWANG / (360.0 * \text{FAVG})$

Common Variables

LB, LE, DT, DF, FLOW, FHIGH, DENSITY, TOTALL, MSAT,
MORB, ALTKM

Input/Output Files

All input and output files are opened and closed in the main program.

Input Files (External)

PARAM.DAT (read in the main program)

The external input file PARAM.DAT consists of two lines having the following format:

PF_SRCH_BAND, R_ARM, ODF_TIME, ODF_FFT
DENSITY, TOTALL, MSAT, MORB, ALTKM

IDFTXY.DAT (read in the subroutine READINDATA)

The external input file IDFTXY.DAT simulates the pre-processed telemetry data stream. It consists of a maximum of 3000 lines with the following format:

TIME, X(I), Y(I), TLG(I), MODE

Output Files

F_FFTX.DAT (written in the subroutine WORK)

The FFT of the x axis gyro rate data is written to file F_FFTX.DAT.

F_FFTY.DAT (written in the subroutine WORK)

The FFT of the y axis gyro rate data is written to file F_FFTY.DAT.

Both F_FFTX.DAT and F_FFTY.DAT have the following format:

FREQUENCY, MODULUS, REAL PART, IMAGINARY PART

Writing to both F_FFTX.DAT and F_FFTY.DAT is controlled by the logical flag ODF_FFT.

F_TXYUV.DAT (written in the main program)

Time domain data is written to file F_TXYUV.DAT. The logical flag ODF_TIME controls writing to F_TXYUV.DAT. The format for F_TXYUV.DAT is:

T, WX, WY, U, V (defined above in variable list).

F_YAWMAN.DAT (written in the main program)

If yaw maneuver calculations are done, the results are written to the file F_YAWMAN.DAT, which has the format:

TMIDPT, TF

(K-1), POLARITY, $360.0 * FAVG$, RNROT, TT

where K runs from 1 to 5 (other variables defined as above in the variable list).

F_RECORD.DAT (written in subroutine ODTF)
See the subroutine ODTF for a description of this file.

Subroutine Calls (in order of calling)

READINDATA (T0,TF,X,Y,TLG,JMODE,LF,M_FLAG)
FBAND (FLOW,FHIGH,TLNGTH,PF_SRCH_BAND)
Subroutine FBAND is called twice, the first time to
return values of FLOW and FHIGH to use in estimating
the number of data points necessary for 3 cycles of the
skiprope, and the second to actually calculate FLOW and
FHIGH for the frequency search band.
WORK (11, X, AMPX, PHASEX, FREQX, IWXMAX, G_FLAGX, AVGX,
ODF_FFT)
WORK (12, Y, AMPY, PHASEY, FREQY, IWYMAX, G_FLAGY, AVGY,
ODF_FFT)
ODTF (T0, TMIDPT, TF, DT, LE, LB, LEB, TLNGTH, AMPX,
FREQX, PHASEX, AMPY, FREQY, PHASEY, FAVG, TAVG,
WK, PSIGN, UMAX, VMAX, FLOW, FHIGH, AVGX, AVGY)

Code Discussion

The parameters NDIM, NFT, and NPNT are set to the
values 3000, 8192, and 7, respectively, the constants
PI, DFR, and RFD are set to 3.1415926, 57.2957795, and
0.017453292, respectively, and the arrays X, Y, and TLG
are dimensioned to NDIM. AMCSMODE is declared as an
integer, REPLY and POLARITY as character variables,
G_FLAGX, G_FLAGY, S_FLAG, M_FLAG, ODF_TIME, and ODF_FFT
as logical variables, and PF_SRCH_BAND, R_ARM, DENSITY,
TOTALL, MSAR, MORB, and ALTKM as reals. The variables
LB, LE, DT, DF, FLOW, FHIGH, DENSITY, TOTALL, MSAT,
MORB, and ALTKM are declared common.

The external file PARAM.DAT is opened as logical
unit 10. The values of PF_SRCH_BAND, R_ARM, ODF_TIME,
ODF_FFT, DENSITY, TOTALL, MSAT, MORB, and ALTKM are
read and PARAM.DAT is closed. All output files are
opened, with F_FFTX.DAT as unit 11, F_FFTY.DAT as unit
12, F_TXYUV.DAT as unit 13, F_YAWMAN.DAT as unit 17,
and F_RECORD.DAT as unit 18.

The subroutine READINDATA is called to read in the
preprocessed telemetry data. (At present, this data is
simulated in the file IDFTXY.DAT.) READINDATA returns
the values of T0, TF, JMODE, LF, M_FLAG, and the arrays
X, Y, and TLG. The variable AMCSMODE is set to the
value of JMODE. If M_FLAG is .FALSE., the operator is
warned that the AMCSMODE changed during the data stream
and the number of data points may be reduced. If the

value of AMCSMODE is either a 0 (indicating an invalid data set) or a 3 (spin case), the operator is alerted and the program aborts.

The yaw maneuver flag S_FLAG is set to .FALSE. The operator is asked whether yaw maneuver calculations are required or not, and is advised that the calculations will not be performed unless requested. Only if the operator replies with 'y' or 'Y' will S_FLAG be set to .TRUE. and yaw maneuver calculations executed.

The average sample time $DT = (TF - T0)/(LF - 1)$ and a preliminary tether length $TLNGTH = 0.5 * (TLG(1) + TLG(LF))$ are calculated. The subroutine FBAND is called (passing this TLNGTH and PF_SRCH_BAND) to return values of FLOW and FHIGH used in estimating the number of data points necessary to comprise 3 skiprope cycles, $LEAST = INT(6.0/(FLOW + FHIGH))$. LEAST is printed to the screen, and the operator is prompted to enter LB, the starting time index, and LE, the last time index. If $LE - LB + 1$ is an even number, set $LE = LE - 1$ so that the total number of time points $LEB = LE - LB + 1$ is odd. (An odd number is necessary for proper use of the HANN window subroutine called by the WORK subroutine.) The tether length TLNGTH is recalculated as $TLNGTH = 0.5 * (TLG(LB) + TLG(LE))$. This value is printed to the screen for the operator's approval. Subroutine FBAND is called again with the new value of TLNGTH and returns the values of FLOW and FHIGH used as the end points of the frequency search band. These values of FLOW and FHIGH are printed to the screen for the operator's approval. (The operator may change the values of TLNGTH, FLOW, and FHIGH if disapproved.) The values of LB, LE, LEB, and TLNGTH are printed to the screen. $DF = 1.0/(NFT*DT)$ (the frequency sample rate), $TSHIFT = DT * (LB - 1)$ (the time shift from T0, the first point in the data buffer, to the time of LB, the start index), and $TMIDPT = T0 + DT * ((LB + LE - 2)/2)$ (the time of the midpoint of the data window) are now calculated.

The WORK subroutine is called twice, once passing the array X and the flag ODF_FFT, and once passing the array Y and the flag ODF_FFT. WORK returns the values of AMPX, PHASEX, FREQX, IWXXMAX, G_FLAGX, and AVGX after the first call, and AMPY, PHASEY, FREQY, IWYMAX, G_FLAGY, and AVGY after the second call. PSIGN is set to the default value of 0.0 (if PSIGN remains as 0.0 then this indicates failure to predict the skiprope frequency). The flags G_FLAGX and G_FLAGY are checked, with 4 resulting cases:

- 1) If both G_FLAGX and G_FLAGY are true, calculate FAVG as the average of FREQX and FREQY, the period TAVG as the reciprocal of FAVG, the constant $WK = 1000.0 * TLNGTH * TAVG / (360 * PI)$, $UMAX = WK * AMPY$, and $VMAX = WK * AMPY$

AMPX. Print the values of AMPX, PHASEX, and FREQX, AMPY, PHASEY, and FREQY, $57.3 * (\text{PHASEX} - \text{PHASEY})$ (the phase difference between x and y), UMAX and VMAX, and TAVG to the screen.

- 2) If G_FLAGX is true and G_FLAGY is false, set FAVG = $\text{FREQX} / \text{TAVG}$, $\text{TAVG} = 1.0 / \text{FAVG}$, $\text{WK} = 1000.0 * \text{TLNGTH} * \text{TAVG} / (360 * \text{PI})$, $\text{VMAX} = \text{WK} * \text{AMPX}$, and $\text{UMAX} = 7777.0$. Print to the screen the warning that the y axis data is suspect, with the calculated frequency outside the search band. (The frequency returned for y is the predicted midpoint of the search band.) This data should not be used without caution. Neither the polarity nor the yaw maneuver calculations are performed. The value of VMAX is printed to the screen.
- 3) If G_FLAGY is true and G_FLAGX is false, set FAVG = $\text{FREQY} / \text{TAVG}$, $\text{TAVG} = 1.0 / \text{FAVG}$, $\text{WK} = 1000.0 * \text{TLNGTH} * \text{TAVG} / (360 * \text{PI})$, $\text{UMAX} = \text{WK} * \text{AMPY}$, and $\text{VMAX} = 7777.0$. Print to the screen the warning that the x axis data is suspect, with the calculated frequency outside the search band. (The frequency returned for x is the predicted midpoint of the search band.) This data should not be used without caution. Neither the polarity nor the yaw maneuver calculations are performed. The value of UMAX is printed to the screen.
- 4) If both G_FLAGX and G_FLAGY are false, set UMAX, VMAX, FAVG, TAVG, WK, and PSIGN to 7777.0 and print to the screen that both axes are bad and offer 3 suggestions for action: 1) look at the time plots of the gyro signals; 2) look at the FFT plots; and 3) widen the search band.

The polarity PSIGN is calculated using the difference between IWXMAX and IWYMAX and comparing to $0.5 * \text{TAVG}$. Two cases are checked: 1) IWYMAX greater than IWXMAX, and 2) IWXMAX greater than IWYMAX. For case 1) TEST is set to $\text{DT} * (\text{IWYMAX} - \text{IWXMAX})$. If TEST is greater than $0.5 * \text{TAVG}$, then PSIGN = -1.0, else PSIGN = 1.0. For case 2) TEST is set to $\text{DT} * (\text{IWXMAX} - \text{IWYMAX})$.

If TEST is greater than $0.5 * \text{TAVG}$, then PSIGN = 1.0, else

PSIGN = -1.0. Print PSIGN to the screen.

If the logical flag ODF_TIME is set to .TRUE., then calculate TWX, TWY, WX, WY, $\bar{U}$, V, and T, and print T, WX, WY, U, and V to the file F_TXYUV.DAT. Note that all of these variables are only calculated if ODF_TIME is true. (As stated in the variables definition section, $\text{TWX} = \cos(2.0 * \text{PI} * \text{FAVG} * \text{DT} * (\text{I} - 1) + \text{PHASEX})$, $\text{TWY} = \cos(2.0 * \text{PI} * \text{FAVG} * \text{DT} * (\text{I} - 1) + \text{PHASEY})$, $\text{WX} = \text{TWX} * \text{AMPX}$, $\text{WY} = \text{TWY} * \text{AMPY}$, $\text{U} = -\text{PSIGN} * \text{WK} * \text{AMPY} * \text{TWX}$, $\text{V} = -\text{PSIGN} * \text{WK} * \text{AMPX} * \text{TWY}$, and $\text{T} = \text{T0} + (\text{LB} + \text{I} - 2) * \text{DT}$.) If the yaw maneuver logical flag S_FLAG is .TRUE., then calculate the number of rotations the orbiter

should execute, $RNROT = (\text{the minimum of } (UMAX, VMAX)) / (2.0 * R_ARM)$. If $PSIGN = 1.0$, set the character variable POLARITY to 'POSITIVE', and if $PSIGN = -1.0$, set POLARITY to 'NEGATIVE'. Calculate the time when the tether is over the orbiter nose, $TXMAX = DT * (IWXMAX - 1)$ (note that this time is a relative time to the beginning of the data window only!). Calculate the time when the tether is over an orbiter wing, $TYMAX = TXMAX + 0.25 * TAVG + TSHIFT + T0$ (note that this is an absolute or mission time quantity). Print the values of the data window midpoint time TMIDPT and the last time in the data buffer TF to both the screen and the file F_YAWMAN.DAT. The time TYMAX must be adjusted to account for the orbiter orientation with respect to the yaw axis. The operator is prompted to input and verify the orbiter yaw axis angle OYAWANG. Calculate the burn time delay $BTDEL = PSIGN * OYAWANG / (360.0 * FAVG)$ and add to TYMAX. Compare the time TYMAX to TF.

If TYMAX is less than or equal to TF, add multiples of the period TAVG to TYMAX ($TYMAX = TYMAX + TAVG$) until TYMAX exceeds TF. For $K = 1$ to 5, calculate the time for the yaw maneuver $TT = TYMAX + (K - 1) * TAVG$, and print both to the screen and the file F_YAWMAN.DAT the revolution label $(K - 1)$, POLARITY, $360.0 * FAVG$, RNROT, and TT.

Regardless of whether the file F_TXYUV.DAT has been printed or not, irrespective of the value of S_FLAG, and for all 4 cases of G_FLAGX and G_FLAGY combinations, call subroutine ODTF and write the file F_RECORD.DAT. (Pass the values of: T0, TMIDPT, TF, DT, LE, LB, LEB, TLNGTH, AMPX, FREQX, PHASEX, AMPY, FREQY, PHASEY, FAVG, TAVG, WK, PSIGN, UMAX, VMAX, FLOW, FHIGH, AVGX, AVGY.) Close all files (logical units 11, 12, 13, 17, and 18).

Subroutine WORK(IOA, ANG, AMP, PHASE, FREQ, ITMAX,
G_FLAG, BIAS, FFT_FLAG)

Definition of Variables/Flags

Variables passed as arguments

IOA - logical unit number for writing output file
unit 11 is for file F_FFTX.DAT, 12 for F_FFTY.DAT
ANG - gyro rate data (either x or y axis)
AMP - amplitude of the gyro rate data at the calculated
skiprope frequency FREQ
PHASE - phase of the skiprope frequency relative to the
beginning of the data window at index LB
FREQ - calculated skiprope frequency
ITMAX - time index of the maximum gyro rate

G_FLAG - logical flag indicating whether the returned value of FREQ is good, i.e., whether FREQ is found within the bounds of the frequency search band
BIAS - mean value of the gyro rate data array ANG after applying Hann window
FFT_FLAG - logical flag controlling the writing of the files F_FFTX.DAT and F_FFTY.DAT; initially set to .TRUE.

Parameters

NDIM - dimension of the arrays ang and aux (set = 3000)
NCDIM - dimension of the complex array awo (set = 8200)
NFT - number of points in the Fourier Transform (set = 8192)
NPNT - number of points used in the least squares curve fitting polynomial (set = 7)

Constants

PI - set to 3.1415926
DFR - conversion factor from radians to degrees (set to 57.2957795)
RFD - conversion factor from degrees to radians (set to 0.017453292)

Common Variables

LB - first time index (of gyro rate data array ang)
LE - last time index (of gyro rate data array ang).
DT - average sample time
DF - frequency sample rate
FLOW - low frequency bound of the frequency search band
FHIGH - high frequency bound of the frequency search band

Other Variables

AUX - gyro rate data array, shifted so first index is 1
AWO - complex data array, used for the Fourier Transform
NTB1 - number of data points in array ANG (LE - LB + 1)
LB1 - index shift used in creating array AUX (1 - LB)
XMAX - maximum value of gyro rate data found in array AUX
IFRST - index of the transformed array AWO corresponding to FLOW (IFRST = 1 + INT(FLOW/DF))
ILAST - index of the transformed array AWO corresponding to FHIGH (ILAST = 1 + INT(FHIGH/DF))
FR - frequency corresponding to index I in transformed array AWO (FR = (I - 1)*DF)
KF - index of maximum modulus of array AWO
XFREQ - array of 7 points, consisting of the moduli of the 7 entries of the array AWO with indices centered about KF, used in the least squares curve fitting of FREQ

PHIMAG - array of 7 points, consisting of the imaginary parts of the 7 entries of the array AWO with indices centered about KF, used in the least squares fitting of PHASEI
 PHREAL - array of 7 points, consisting of the real parts of the 7 entries of the array AWO with indices centered about KF, used in the least squares fitting of PHASER
 FQ_P0 - interpolated index value returned by the curve fitting subroutine LSCF
 PHASEI - imaginary part of PHASE, interpolated by LSCF
 PHASER - real part of PHASE, interpolated by LSCF
 SCALE - scaling factor to give transformed data in units of deg/sec and represent actual rate data

Output Files

F_FFTX.DAT, F_FFTY.DAT

File F_FFTX.DAT has logical unit number 11 (stored in IOA), and F_FFTY.DAT has logical unit number 12. For long tether lengths (small skiprope frequencies), 1006 lines are printed; for short tether lengths (larger skiprope frequencies), 336 lines are printed. Each line has the format:

FR, XMAX, REAL(AWO(I)), AIMAG(AWO(I))
 (Here XMAX = SCALE * modulus of AWO(I))

Subroutines Calls (in order of calling)

HANN (NTB1, AUX, BIAS)
 FOUR1 (AWO, NFT, 1)
 LSCF (FQ_P0, XMAX, XFREQ, 1, G_FLAG)
 G_FLAG is set after this first call to LSCF (with the 1 in the 4th argument). If G_FLAG is .TRUE., then LSCF is called twice more to interpolate the imaginary and real parts of PHASE:
 LSCF (FQ_P0, PHASEI, PHIMAG, 2, G_FLAG)
 LSCF (FQ_P0, PHASER, PHREAL, 2, G_FLAG)

Code Discussion

The values of the gyro rate data array ANG, logical unit indicator IOA, and file print flag FFT_FLAG are passed in the calling statement, as are the common variables LB, LE, DT, DF, FLOW, and FHIGH. The parameters NDIM, NCDIM, NFT, and NPNT are set to the values 3000, 8200, 8192, and 7, respectively, and the constants PI, DFR, and RFD to 3.1415926, 57.2957795, and 0.017453292, respectively. The real array AUX is dimensioned to

NDIM, the real arrays XFREQ, PHIMAG, and PHREAL to NPNT, the complex array AWO to NCDIM, and G_FLAG and FFT_FLAG are declared logical variables.

The number of data points NTB1 in the array ANG (NTB1 = LE - LB + 1) and the first index shift LB1 (LB1 = 1 - LB) are calculated. Letting the index I range from LB to LE, the new index IL = I + LB1 ranges from 1 to NTB1. Set the array AUX(IL) = ANG(I), so AUX is the same array as ANG, but with starting index 1 rather than LB. Calculate the mean of the array AUX (and thus of ANG) and apply the HANN window to AUX by calling the subroutine HANN.

Make the complex array AWO by setting the real parts of the first NTB1 entries of AWO equal to the corresponding entry in AUX, with the imaginary parts set to 0.0, and padding the rest of AWO with zeroes. Find the Fourier Transform of AWO using the FFT routine FOUR1 (version supplied by "Numerical Recipes").

Search for the maximum modulus of the transformed array AWO. First, set XMAX = 0.0, and calculate the frequency indices IFRST = 1 + INT(FLOW/DF) and ILAST = 1 + INT(FHIGH/DF). For I ranging from IFRST to ILAST, calculate FR = (I-1)*DF, and check to see if the modulus

of AWO(I) (CABS(AWO(I))) is greater than XMAX; if so, set XMAX = CABS(AWO(I)), KF = I (save the index of the maximum found), and FREQ = FR (save the frequency of the maximum).

Once the search is completed and the maximum known, interpolate to find the best quadratic fitting the 7 points centered on the maximum. The array XFREQ holds the values of the moduli of the 7 points, the array PHIMAG holds the imaginary parts of the 7 points, and the array PHREAL holds the real parts of the 7 points. Call the curve fitting subroutine LSCF (passing XFREQ) to calculate the interpolated frequency index FQ_P0 and the modulus XMAX at FQ_P0, and set G_FLAG to true or false. If G_FLAG is true, then call LSCF again (passing PHIMAG) to find PHASEI, the imaginary part of PHASE, and call LSCF a third time (passing PHREAL) to find PHASER, the real part of PHASE. The maximum frequency is FREQ = FREQ + DF * FQ_P0. If G_FLAG is false, then set KF to the index of the frequency search band midpoint, XMAX = CABS(AWO(KF)), PHASEI = AIMAG(AWO(KF)), PHASER = REAL(AWO(KF)), and FREQ = DF * (KF - 1). Calculate the scaling factor SCALE = 4.0/(NTB1 - 1), the scaled maximum modulus AMP = SCALE * XMAX, the PHASE = -ATAN2(PHASEI, PHASER), and the time index of the maximum frequency ITMAX = INT((1.0/(FREQ*DT))*(1.0-PHASE/(2.0*PI)) + 0.5) + 1. If ITMAX is greater than one period, subtract one period from ITMAX, i.e., if ITMAX*DT is greater than (1.0/FREQ), then ITMAX = ITMAX -

INT(1.0/(FREQ*DT)).

If FFT_FLAG is true, print to the file indicated by the logical unit number stored in IOA. Print 1006 lines, unless FREQ is greater than 0.0035, in which case only print 336 lines. For I ranging from 1 to either 336 or 1006, calculate $FR = (I-1)*DF$, $XMAX = SCALE*CABS(AWO(I))$, and write FR, XMAX, REAL(AWO(I)), and AIMAG(AWO(I)) to the output file.

Subroutine HANN (LA, A11, BIAS)

Definition of Variables

Variables passed as arguments

LA - number of data points in array A11 (equals NTB1)

A11 - data array (ANG or AUX) which Hann window is applied to

BIAS - mean of array A11 (after windowing)

Constants

PI - set to 3.1415926

Other Variables

HW - array holding the calculated discrete Hann window

ITM - index of midpoint of array A11

RM - ITM as a real variable

Subroutine Calls

MEAN (LA, A11, BIAS)

Code Discussion

The array A11, and LA, the length of A11, are passed in the calling statement. The constant PI is set to 3.1415926, and the array HW is dimensioned to 3000. The index of the midpoint of A11 is calculated, $ITM = (LA - 1)/2$, and converted to a real value RM. For index IT ranging from -ITM to ITM, calculate index $I = 1 + IT + ITM$ and $HW(I) = 0.5 * (1.0 - \cos(\pi * IT)/RM)$, and apply the Hann window HW to the array A11, $A11(I) = A11(I) * HW(I)$. Call subroutine MEAN to find the mean value BIAS of the windowed array A11, and subtract the windowed BIAS from the windowed array, $A11(I) = A11(I) - HW(I) * BIAS$.

Subroutine MEAN (LA, A22, SA)

Definition of Variables

Variables passed as arguments
LA - length of the data array A22
A22 - data array
SA - mean value of array A22

Code Discussion

The values of the array A22, and LA, the length of array A22, are passed by the calling statement. SA is the sum of the values of the individual entries of A22, divided by LA.

Subroutine FOUR1 (DATA, NN, ISIGN)

Subroutine FOUR1 is the standard FFT routine found in "Numerical Recipes".

Definition of Variables

Variables passed as arguments
DATA - data array (complex, but converted to a real array of double length)
NN - number of points in the Fourier Transform
ISIGN - +1 indicates forward transform, -1 inverse

Other Variables
WR, WI, WPR, WPI, WTEMP, THETA - all double precision variables used in the usual array shuffling procedures

Code Discussion

Subroutine FOUR1 utilizes the array shuffling procedure common to most FFT routines. For more details, consult "Numerical Recipes", or other sources that discuss FFT routines at length.

Subroutine LSCF (P0, FMAX, U_IN, IOPT, G_FLAG)

Definition of Variables/Flags

Variables passed as arguments

P0 - interpolated location of maximum
FMAX - value of the maximum located at P0
U_IN - input ordinate array (length of 7)
IOPT - action option, either 1 or 2
 IOPT = 1: Find P0 and compute maximum at P0
 IOPT = 2: Only compute value at P0
G_FLAG - indicates data validity, TRUE if the peak is
 inside the search zone, FALSE if outside

Other Variables

US17 - sum of U_IN(1) and U_IN(7)
US35 - sum of U_IN(3) and U_IN(5)
COF1 - constant term in fitting quadratic
COF2 - coefficient of linear term in fitting quadratic
COF3 - coefficient of square term in fitting quadratic

Code Discussion

Subroutine LSCF does a least squares curve fit of a quadratic to 7 data points sampled at integral intervals. The indices of the 7 points range from -3 to +3. The quadratic is $F(P) = COF1 + COF2*P + COF3*P**2$, the max occurs at $P0 = -COF2/(2*COF3)$, and the maximum value is $F(P0) = COF1 - (COF2**2)/(4.0*COF3)$. The 3 coefficients are computed by multiplying the 3x7 matrix

-8	12	24	28	24	12	8
-9	-6	-3	0	3	6	9
5	0	-3	-4	-3	0	5

(each term is divided by 84)

times the array U_IN (7 entries in the array), with the results $COF1 = \text{first row} \times U_IN$, $COF2 = \text{second row} \times U_IN$, and $COF3 = \text{third row} \times U_IN$. The array U_IN and the option variable IOPT are passed by the calling statement. To further use the symmetry of the matrix, $US17 = U_IN(1) + U_IN(7)$ and $US35 = U_IN(3) + U_IN(5)$ are created. COF1, COF2, and COF3 are calculated as described above. COF3 is checked to be sure that it does not equal 0.0 (equaling 0.0 would prevent calculation of the maximum $F(P0) = COF1 - (COF2**2)/(4.0*COF3)$); if COF3 = 0.0 then G_FLAG is set to false and control returns to the calling subroutine WORK.

For IOPT option 1, calculate $P0 = -0.5*COF2/COF3$, $FMAX = (COF1 + 0.5*P0*COF2)/84$, and set G_FLAG to true. To check if the calculated P0 is valid, compare the absolute value of P0 to 3; if ABS(P0) greater than 3, then set G_FLAG to false. For IOPT option 2, calculate $FMAX = (COF1 + P0*(COF2 + P0*COF3))/84$.

Subroutine FBAND (FL, FH, TLKM, PF)

Definition of Variables

Variables passed as arguments

FL - low frequency bound of the frequency search band
FH - high frequency bound of the frequency search band
TLKM - tether length in kilometers
PF - percentage used to compute frequency search band

Common Variables

DENSITY - tether density in kg/km (set to 8.35 kg/km)
TOTALL - total tether length (set to 22.0 km)
MSAT - satellite mass in kg (set to 510.0 kg)
MORB - orbiter mass in kg (set to 100000.0 kg)
ALTKM - orbit altitude in km (set to 325.0 km)

Other Variables

FC - center frequency of the frequency search band =
 $0.5 * \text{SQRT}(\text{CK} * \text{MSTAR} / \text{TLKM})$
 (estimate of the skiprope frequency based on the
 tether parameters listed as common variables and
 the average tether length TLKM)
OMSQ - orbit rate squared ($\text{OMSQ} = \text{ORBRATESQ}(\text{ALTKM})$)
CK - working variable = $3.0 * \text{OMSQ} / \text{DENSITY}$
MO - sum of orbiter mass and tether mass (but not the
 satellite mass!) $\text{MO} = \text{MORB} + \text{TOTALL} * \text{DENSITY}$
Q - working variable = $0.5 * \text{DENSITY} * \text{TLKM}$
MSTAR - working variable = $((\text{MO} - \text{Q}) * (\text{MSAT} + \text{Q})) / (\text{MO} + \text{MSAT})$
DF - fraction of FC to be subtracted from FC to create
 FL and added to FC to create FH ($\text{DF} = \text{FC} * \text{PF} / 100.0$)

Function Call

REAL FUNCTION ORBRATESQ (ALTKM)

Parameters

GM - acceleration due to gravity, in meters/sec**2 (set
 to 9.81098)
RE - radius of earth in km (set to 6378.17)

Other Variables

R - $\text{GM} / (1000.0 * \text{RE})$
ORBRATESQ - $\text{R} / (1.0 + \text{ALTKM} / \text{RE}) ** 3$

Code Discussion

The values of the average tether length in km,

TLKM, and the percentage around the estimated skiprope frequency, PF, are passed as arguments of the calling statement, and tether/orbiter parameters DENSITY, TOTALL, MSAT, MORB, and ALTKM are passed as common variables. Using equations derived from the dynamical analysis of the skiprope frequency vs tether length, the estimated skiprope frequency FC is calculated (see the variable list for the equations used). Search band bounds FL and FH are computed from FC by using the percentage PF, $FL = FC - FC * PF / 100.0$ and $FH = FC + FC * PF / 100.0$.

Subroutine READINDATA (T0,TF,X,Y,TLG,MODE,LF,MFLAG)

Definition of Variables/Flags

Variables passed as arguments

T0 - time tag for first point in buffer

TF - time tag for last point in buffer

X(I) - x axis gyro data array (deg/sec)

Y(I) - y axis gyro data array (deg/sec)

TLG(I) - tether length array (kilometers)

MODE - integer variable, signifying the amcsmode for the last time point 'TF'

amcsmode = 0 indicates no valid data

amcsmode = 1 indicates passive case

amcsmode = 2 indicates yaw hold

amcsmode = 3 indicates spin case

LF - number of time points stored in each array

MFLAG - logical flag to denote if the amcsmode changed from 1 or 2 to a 0 or 3 between the times of T0 and TF

Other Variables

JMODE - amcsmode of time tag T0

External Input File

IDFTXY.DAT

The external input file IDFTXY.DAT simulates the preprocessed telemetry data stream. Each line has the following format:

TIME, X(I), Y(I), TLG(I), MODE

Code Discussion

The external input file IDFTXY.DAT is opened as logical unit 10 and the first line read, with the time recorded as T0 and the MODE value as JMODE. A read loop is entered, and lines will be read as long as the

MODE remains unchanged or if the MODE only changes from 1 to 2 or 2 to 1, until the end of the file. If the MODE changes other than from 1 to 2 or 2 to 1, then reading stops, and MFLAG is set to false. The last line read, whether or not the end of file is reached, has the time recorded as TF, the MODE value returned to the main program, and is the point where LF is calculated. The file IDFTXY.DAT is then closed.

Subroutine ODTF (TO, TM, TF, DT, LE, LB, LEB, TL, AX,
FX, PX, AY, FY, PY, FA, TA, WK, PSIGN,
UMAX, VMAX, FL, FH, AVGX, AVGY)

Definition of Variables

Variables passed as arguments

TO - first time point in buffer (mission elapsed time)
 TM - time at midpoint of data window (met)
 TF - last time point in buffer (met)
 DT - sample time (sec)
 LE - index of last point in data window
 LB - index of first point in data window
 LEB - number of points used in data window
 TL - tether length used for this data window (km)
 AX - peak magnitude of x axis gyro rate (deg/sec)
 FX - calculated skiprope frequency in x axis (hz)
 PX - phase angle in x axis
 AY - peak magnitude of y axis gyro rate (deg/sec)
 FY - calculated skiprope frequency in y axis (hz)
 PY - phase angle in y axis
 FA - average frequency with G_FLAG set to true
 TA - average period, reciprocal of FA
 WK - conversion factor from rate data to skiprope amplitude (meter-sec/deg)
 PSIGN - skiprope polarity w.r.t Z LVLH axis
 UMAX - maximum in plane midnode skiprope amplitude
 VMAX - maximum out of plane midnode skiprope amplitude
 FL - lower boundary of the frequency search band (hz)
 FH - upper boundary of the frequency search band (hz)
 AVGX - computed mean of the Hann-windowed x axis gyro rate data
 AVGY - computed mean of the Hann-windowed y axis gyro rate data

Output File

F_RECORD.DAT (with logical unit number 18)

The output file F_RECORD.DAT has 6 lines with the following format:

TM
LB, LE, LEB
TO, TF, DT, TL, WK, PSIGN
AX, FX, PX, AY, FY, PY
FA, TA, UMAX, VMAX, FL, FH
AVGX, AVGY

Code Discussion

The variables in the variable list are passed as arguments and written to logical unit 18 (output file F_RECORD.DAT) in the format described above.

Quick Reference

Required External Files

PARAM.DAT (read in the main program)

The external input file PARAM.DAT consists of two lines having the following format:

PF_SRCH_BAND, R_ARM, ODF_TIME, ODF_FFT
DENSITY, TOTALL, MSAT, MORB, ALTKM

IDFTXY.DAT (read in the subroutine READINDATA)

The external input file IDFTXY.DAT simulates the pre-processed telemetry data stream. It consists of a maximum of 3000 lines with the following format:

TIME, X(I), Y(I), TLG(I), MODE

Operator Input Prompts (in order of appearance in the main program)

Prompt: Yaw maneuver calculations will not be performed unless requested. Type yes (y) to calculate, no (n) otherwise.

Prompt: DT is (DT value), is this okay to use - yes (y)

or no (n)

If no, then prompt: Enter DT in seconds

Prompt: Recommend using at least (LEAST value) points for 3 data cycles. Last point in buffer is (LF). What is the starting time index?

After receiving the start time index LB, the prompt continues with: What is the last time index?

Prompt: Tether length is (TLNGTH value) kilometers - ok to use, reply with a yes (y) or a no (n)

If no, then prompt: Enter the tether length in kilometers, and then repeat original tether length prompt.

Prompt: FLOW & FHIGH = (FLOW, FHIGH values)

Are these bounds okay to use - y or n

If no, then prompt: Enter the two values in hz, and then repeat original frequency bounds prompt.

Prompt: Enter orbiter yaw axis angle in degrees

Verification prompt: Orbiter nose is ___ degrees wrt X-LVLH axis. Is this correct (Y or N)?

If no, repeat the original orbiter yaw axis angle prompt.

Warnings/Abort Situations

If mode flag M_FLAG is false, the following warning is printed to the screen:

WARNING

AMCSMODE STATUS CHANGED DURING THIS DATA STREAM
NUMBER OF POINTS REDUCED - DATA SET HAS SAME MODE
WILL SET AMCSMODE TO LAST READING

If AMCSMODE = 0, then print this warning:

AMCSMODE INDICATES NO VALID DATA - PROGRAM ABORTS

The following warning is printed if AMCSMODE = 3

AMCSMODE INDICATES SPIN CASE - PROGRAM ABORTS

Warning for G_FLAGX false and G_FLAG true:

Y AXIS DATA IS SUSPECT - FREQ OUT OF BAND
FREQUENCY RETURNED IS PREDICTED MIDPOINT
DATA SHOULD NOT BE USED WITHOUT CAUTION
NEITHER POLARITY NOR YAW MANEUVER CALCULATIONS ARE
PERFORMED

Warning for G_FLAGX false and G_FLAG true:

X AXIS DATA IS SUSPECT - FREQ OUT OF BAND
FREQUENCY RETURNED IS PREDICTED MIDPOINT
DATA SHOULD NOT BE USED WITHOUT CAUTION
NEITHER POLARITY NOR YAW MANEUVER CALCULATIONS ARE
PERFORMED

Warning for both G_FLAGX and G_FLAG false:

BOTH AXES ARE BAD ***** 3 SUGGESTIONS

- 1) SUGGEST LOOK AT TIME PLOTS OF GYRO SIGNALS.
- 2) SUGGEST MAKE FFT PLOTS AND LOOK AT DATA.
- 3) SUGGEST WIDENING SEARCH BAND.

If the curve fitting subroutine LSCF calculates the quadratic coefficient COF3 = 0.0, print this warning:

***** WARNING *****

QUADRATIC COEFFICIENT EQUALS ZERO - CANNOT COMPUTE A
MAXIMUM FREQUENCY VALUE.

TEST PLAN FOR THE BACKUP SKIPROPE OBSERVER

UNIVERSITY OF NEW ORLEANS

Steve Rodrigue, George Ioup, Juliette Ioup, Abolfazl Amini

24 DEC 1991

INTRODUCTION

This document describes the Test Plan and Procedures for evaluating the Frequency Domain Skiprope Observer. The plan is divided into two parts, one for Station 2 conditions, and the second for Station 1 conditions. No concrete performance requirements exist at Station 1, because the main focus of the Observer is to support a yaw maneuver at Station 2. Nevertheless, two tests using simulations at Station 1 are included to demonstrate Observer performance. Additional Station 1 tests using model test signals are also included to demonstrate and/or define Observer performance boundaries at Station 1.

The test cases enumerated in the ECR, using both model gyro signal plus actual simulation data, will be fully documented with input data and filter output results. These cases should be used to verify observer code whenever the code is transferred to a different computer system.

This Test Plan calls for using simulated gyro noise. The noise source to be used for all testing is a portable random number generator as documented in Reference _____ and included as Appendix 2.A herein. The model for generating a Gaussian distribution for noise is also a part of Appendix 2.A.

PART I - STATION 2 CONDITIONS: (2.4 km TETHER LENGTH)

This part is divided into four Test Groups:

- A. Six cases using model gyro signals per ECR
- B. Three cases using simulation data per ECR
- C. Systematic error testing without noise (using model gyro signals)
- D. Systematic error testing with noise (using model gyro signals)

The purpose of the first two groups is to establish test case results for code transfer as well as prove performance of the Observer. Where noise is modelled using the random number generator, documentation of these cases will include the initial value of the random number seed. Users of the test plan

- should use any or all of the fully documented cases to verify the code -
results should vary from the results documented only to within expected
roundoff error. Users should also vary the initial value of the random number
seed to fully statistically test the Observer.

Cases in Group A will verify the ECR requirements in paragraph 3.2.(1)
and 3.2.(2), i.e., the error of the angular rate amplitude shall not exceed 2%,
- with a maximum total phase error of 25 degrees after 15 minutes from the last
time point used in the data window.

- Cases 1 and 2 in Group B use simulation data to verify that the secondary
Observer meets the performance requirements for the primary Observer.
Amplitude and phase errors will be measured on a root-mean-square basis as
outlined in the ECR, paragraphs 5.3.2 and 5.3.1.

- Case 3 of Group B will be tested the same as cases 1 and 2; however, the
conditions inherent to this case violate the constraints and limitations
enumerated in the ECR. This case is included to demonstrate trends in
degradation for highly transient conditions.

- Cases in Group C are used to best define the ultimate performance of the
Observer under ideal (noise-free) conditions. Model gyro signals are inputted
to the Observer for a range of frequencies at Station 2 and for various
pairings of skip rope and pendulous phases. Percentage errors between the input
values and output values of the skip rope amplitude, frequency, and phase are
reported.

- Cases in Group D are used to define the expected performance in a noisy
environment. For each case in Group C, 50 different noisy signals are
generated (using the portable random number generator and the Box-Muller
algorithm to generate Gaussian distributed noise described in Appendix 2.A).
- The maximum amplitude, frequency, and phase errors found in the set of 50 noise
runs are reported, as well as the average errors over the 50 runs. For each
frequency tested (11 total), the largest maximum and largest average errors are
also reported.

A: STANDARDIZED TEST CASES

These test cases are specified in the ECR and have a model gyro signal of the following form:

$$N(T) + A0 + A1 \cdot \cos(2 \cdot \pi \cdot F1 \cdot (I-1) \cdot DT + \text{PHI1}) + A2 \cdot \cos(2 \cdot \pi \cdot F2 \cdot (I-1) \cdot DT + \text{PHI2})$$

where $N(T)$ = GAUSSIAN DISTRIBUTED NOISE WITH SIGMA = $2.8E-04$ DEG/S

A0 = ORB RATE (DEG/S)

A1 = SKIPROPE AMPLITUDE (DEG/S)

F1 = SKIPROPE FREQUENCY (HZ)

PHI1 = SKIPROPE PHASE (DEG)

A2 = PENDULOUS AMPLITUDE (DEG/S)

F2 = PENDULOUS FREQUENCY (HZ)

PHI2 = PENDULOUS PHASE (DEG)

DT = SAMPLE TIME (SEC)

Input data files to the Observer must be in the following format:

TIME, X GYRO SIGNAL, Y GYRO SIGNAL, TETHER LENGTH, AMCSMODE

Both the x gyro signal and y gyro signal are of the form detailed above. Please note that the x and y gyro signals should be generated independently, albeit concurrently, and have distinct Gaussian distributed noise terms (as detailed explicitly in the discussion of the noise generation found in Appendix 2.A). At Station 2 the tether length is 2.4 km, and amcsmode can be either 1 or 2. The time values must be spaced at the sample rate, preferably 1.024 s. Users may write their own signal generating programs, or they may use the program CREATE.FOR listed in Appendix 2.A.

The following table is the list of values used to generate the model gyro signals. Please note the following:

- 1) The skiprope amplitude and frequency values, and the pendulous amplitude values are used for both the x and y signals.
- 2) The orb rate is added only to the y gyro signal ($A0 = 0.0$ for the x gyro signal).
- 3) Skiprope phase is the y axis value. The x axis value is phase + 90.0.
- 4) Pendulous phase is the x axis value. The y axis value is phase - 90.0.
- 5) Gaussian distributed standard deviation noise of $2.8E-03$ deg/s is 10 times the expected noise sigma of $2.8E-04$ deg/s.
- 6) All test cases should use data lengths of at least 3 skiprope periods and a sample time of 1.000 s or 1.024 s (one skiprope period is calculated as $1.0 / (\text{freq} \times \text{sample time})$).
- 7) Pendulous frequency = 0.03125 Hz

- 8) Cases with non-zero noise should use 10 runs each. The averages of the 10 runs constitute the output.

CASE NO.	NOISE	ORB RATE	AMP	SKIPROPE FREQ	PHASE	PENDULOUS AMP	PENDULOUS PHASE
1	0.0	0.06	0.02	0.0054	163.0	0.0	10.0
2	2.8E-03	0.06	0.02	0.0054	163.0	0.0	10.0
3	0.0	0.06	0.02	0.0046	-40.0	0.5	60.0
4	2.8E-03	0.06	0.02	0.0046	-40.0	0.5	60.0
5	2.8E-03	0.06	0.15	0.0054	70.0	0.0	50.0
6	2.8E-03	0.06	0.15	0.0054	70.0	0.5	50.0

- Appendix 2.B lists tables of results for all six cases and 40 initial values of the random number seed (4 test cases with non-zero noise x 10 noise runs each).

B: SIMULATION RUNS (VERIFICATION MATRIX)

For this group of the simulations, the user should use the option to print the calculated rates in the Observer program (UNOMSC.FOR), i.e., set ODF_TIME to true. A root_mean_square comparison of the skiprope amplitudes and phases is then performed between the original simulation data and the data generated by the Observer. A sample comparison program is listed in Appendix 2.C.

CASE	SKIPROPE (IN PLANE x OUT OF PLANE)	TETHER LENGTH (km)
1	20 x 20	2.4
2	60 x 60	2.4
3	80 x 40	2.4

Notes:

- 1) Data length should be at least 3 skiprope periods with a sample time of 1.000 s or 1.024 s.
- 2) These 3 simulations are required by the ECR. The observer should work properly given any valid simulation, i.e., a simulation without spin.

C: SYSTEMATIC ERROR TESTING ACROSS A RANGE OF SKIPROPE
FREQUENCIES AND SKIPROPE / PENDULOUS PHASE PAIRINGS
NOISE-FREE CASE

Model gyro signals are of the same form as detailed in section A. These tests are of the gyro signal itself, so only one axis is necessary (all parameters are used on this one axis). The programs listed in Appendix 2.D automate the procedure of creating the signals and running the essentials of the Observer by incorporating various loops into the body of the program to eliminate user input in creating data files and/or running the Observer.

For each frequency in the range (0.0045 - 0.0055 Hz), at intervals of 0.0001 Hz (a total of 11 frequencies), run the following tests:

Parameters:

- orb rate = 0.065 deg/s
- skiprope amp = 0.02 deg/s
- pendulous frequency = 0.03125 Hz
- pendulous amplitude = 0.5 deg/s
- data length = at least three periods of data
- noise = 2.8E-03 deg/s
- sample rate = 1.024 s or 1.000 s

- 1) Vary the skiprope phase from -180.0 deg to 180.0 deg in increments of 10 deg. For each skiprope phase vary the pendulous phase from -180.0 deg to 180.0 deg in increments of 10 deg (a total of $37 \times 37 = 1369$ cases).
- 2) In each case record the per cent errors in the calculated skiprope amplitude, frequency, and phase.
- 3) Find the maximum amplitude, frequency, and phase per cent errors and the associated skiprope and pendulous phases for each.
- 4) Plot error surfaces of the amplitude, frequency, and phase errors (a total of 33 plots - 3 plots for each frequency x 11 frequencies).

D: SYSTEMATIC ERROR TESTING ACROSS A RANGE OF SKIPROPE
FREQUENCIES AND SKIPROPE / PENDULOUS PHASE PAIRINGS
NOISE CASE

Model gyro signals are of the same form as detailed in section A. These tests are of the gyro signal itself, so only one axis is necessary (all parameters are used on this one axis). The programs listed in Appendix 2.D automate the procedure of creating the signals and running the essentials of the Observer by incorporating various loops into the body of the program to eliminate user input in creating data files and/or running the Observer.

- For each frequency in the range (0.0045 - 0.0055 Hz), at intervals of 0.0001 Hz (a total of 11 frequencies), run the following tests:

Parameters:

orb rate = 0.065 deg/s
skiprope amp = 0.02 deg/s
pendulous frequency = 0.03125 Hz
pendulous amplitude = 0.5 deg/s
data length = at least three periods of data
noise = 2.8E-03 deg/s
sample rate = 1.024 s or 1.000 s

- 1) For each of the 1369 phase relationship cases listed in C.1 for the noise-free case, run 50 noise runs (Gaussian distributed noise).
- 2) In each case record the average and maximum per cent errors in the calculated skiprope amplitude, frequency, and phase.
- 3) Find the maximum per cent errors in the amplitude, frequency, and phase and the associated skiprope and pendulous phases for each.
- 4) Find the largest maximum errors in the amplitude, frequency, and phase and the associated skiprope and pendulous phases for each.
- 5) Plot representative samples of the surfaces generated in part b).

PART II - STATION 1 CONDITIONS: (20.0 km TETHER LENGTH)

- This part is divided into three Test Groups:

- A. Two cases using simulation data per ECR
- B. Systematic error testing without libration component - both noise-free and noise cases (using model gyro signals)
- C. Systematic error testing with libration component - both noise-free and noise cases (using model gyro signals)

- Cases 4 and 5 in Group A use simulation data to verify that the secondary Observer meets the performance requirements for the primary Observer. Amplitude and phase errors will be measured on a root-mean-square basis as outlined in the ECR, paragraphs 5.3.2 and 5.3.1. (Note: The numbering of the simulation cases follows the convention of the ECR, which has simulations from both stations 2 and 1 in one table and numbered sequentially - cases 1, 2, and 3 at station 2 cases 4 and 5 at station 1.)

Cases in Group B are used to define the expected performance of the Observer at Station 1 without the influence of a libration component. Both the noisy and ideal (noise-free) environments are considered.

Cases in Group C are used to define the expected performance of the Observer at Station 1 with the influence of a libration component. Both the noisy and ideal (noise-free) environments are considered.

For both Groups B and C, runs are performed for all cases with data lengths of 2, 3, and 4 skiprope periods. Noise runs are performed with both 1 sigma and 3 sigma Gaussian distributed noise standard deviations.

A: SIMULATION RUNS (VERIFICATION MATRIX)

For this group of the simulations, the user should use the option to print the calculated rates in the Observer program (UNOMSC.FOR), i.e., set ODF_TIME to true. A root mean square comparison of the skiprope amplitudes and phases is then performed between the original simulation data and the data generated by the Observer. A sample comparison program is listed in Appendix 2.C.

CASE	SKIPROPE (IN PLANE x OUT OF PLANE)	TETHER LENGTH (km)
4	80 x 40	20.0
5	80 x 80	20.0

Notes:

- 1) Data length should be at least 3 skiprope periods with a sample time of 1.000 s or 1.024 s.
- 2) These 2 tests are required by the ECR. The observer should be able to work properly given any valid simulation, i.e., a simulation without spin.

B: SYSTEMATIC ERROR TESTING OF MODEL SIGNALS WITHOUT LIBRATION USING DATA LENGTHS OF 2, 3, OR 4 SKIPROPE PERIODS

Model gyro signals are of the same form as detailed in section A, part I. The CREATE.FOR program listed in Appendix 2.A can be used to generate signals for this group. Appendix 2.D lists programs that automate the data file generation and Observer testing for the cases in this group.

For each of the desired data lengths, data files should be generated with the following values:

PARAMETERS:

		LIBRATION	SKIPROPE	PENDULOUS	
		-----	-----	-----	
AMPLITUDE	(X AXIS)	0.004	0.0034	0.05	
	(Y AXIS)	0.004	0.0034	0.05	(units = deg/s)
FREQUENCY	(X AXIS)	1/2713	0.0019	0.089	
	(Y AXIS)	1/3132	0.0019	0.089	(units = Hz)
PHASE	(X AXIS)	varies	varies	0.0	
	(Y AXIS)	varies	varies	-90.0	(units = deg)

SIGMA = $2.8E-04$ deg/s

ORB RATE = 0.065 deg/s

DT = 1.024 s or 1.000 s

1) For the model signal without libration component vary the skiprope phase from -180.0 deg to 180.0 deg in increments of 10 deg (a total of 37 cases).

2) In each case record the per cent errors in the calculated skiprope amplitude, frequency, and phase.

3) Find the maximum amplitude, frequency, and phase per cent errors and the associated skiprope phases for each.

4) Plot error curves of the amplitude, frequency, and phase errors (a total of 3 plots).

Do the following steps for Gaussian distributed noise signals, using noise = $1 \times \text{sigma} = 2.8E-04$ and noise = $3 \times \text{sigma} = 8.4E-03$:

5) For each of the 37 phase relationship cases listed in 1) for the noise-free case, run 50 noise runs (Gaussian distributed noise).

6) In each case record the average and maximum per cent errors in the calculated skiprope amplitude, frequency, and phase.

7) Plot representative samples of the curves generated in part 6).

C: SYSTEMATIC ERROR TESTING OF MODEL SIGNALS WITH LIBRATION
USING DATA LENGTHS OF 2, 3, OR 4 SKIPROPE PERIODS

Model gyro signals are of the same form as detailed in section A, part I, with the addition of a libration component term of the form:

$$ALIB * \cos(2 * \pi * FLIB * (I-1) * DT + PHLIB)$$

where ALIB = LIBRATION AMPLITUDE
FLIB = LIBRATION FREQUENCY
DT = SAMPLE RATE
PHLIB = LIBRATION PHASE

Appendix 2.A lists a program (CRELIBR.FOR) that generates a model gyro signal with a libration component. Appendix 2.D lists programs that automate the signal generation and Observer evaluation for the cases in this group.

For each of the desired data lengths, data files should be generated with the following values:

PARAMETERS:		LIBRATION	SKIPROPE	PENDULOUS	
		-----	-----	-----	
AMPLITUDE	(X AXIS)	0.004	0.0034	0.05	(units. = deg/s)
	(Y AXIS)	0.004	0.0034	0.05	
FREQUENCY	(X AXIS)	1/2713	0.0019	0.089	(units = Hz)
	(Y AXIS)	1/3132	0.0019	0.089	
PHASE	(X AXIS)	varies	varies	0.0	(units = deg)
	(Y AXIS)	varies	varies	-90.0	
SIGMA = 2.8E-04 deg/s					
ORB RATE = 0.065 deg/s					
DT = 1.024 s or 1.000 s					

- 1) For the model signal with libration component vary the skiprope phase from -180.0 deg to 180.0 deg in increments of 10 deg. For each skiprope phase, vary the libration phase from -180.0 deg to 180.0 deg (a total of 1369 cases 37 skiprope phases x 37 libration phases).
- 2) In each case record the per cent errors in the calculated skiprope amplitude, frequency, and phase.
- 3) Find the maximum amplitude, frequency, and phase per cent errors and the associated skiprope and libration phases for each.
- 4) Plot error surfaces of the amplitude, frequency, and phase errors (a total of 3 plots).

Do the following steps for Gaussian distributed noise signals, using noise = 1 x sigma = 2.8E-04 and noise = 3 x sigma = 8.4E-03:

- 5) For each of the 1369 phase relationship cases listed in 1) run 50 noise runs (Gaussian distributed noise).
- 6) In each case record the average and maximum per cent errors in the calculated skiprope amplitude, frequency, and phase.
- 7) Find the maximum average per cent errors in the amplitude, frequency, and phase and the associated skiprope and libration phases for each.
- 8) Find the largest maximum errors in the amplitude, frequency, and phase and the associated skiprope and libration phases for each.
- 9) Plot representative samples of the surfaces generated in part 6).

APPENDIX 1

```

C      NAME IS UNOMSC.FOR (BACK UP SKIPROPE OBSERVER)
C      THIS VERSION IS COMBINED FROM UNO AND MSFC
C      DATE IS SEPTEMBER 1991
C
C      INTEGER NDIM, NFT, NPNT, AMCSMODE
C      PARAMETER (NDIM= 3000, NFT=8192, NPNT=7)
C      PARAMETER (PI=3.1415926, DFR=57.2957795, RFD=0.017453292)
C      REAL*4 X(NDIM), Y(NDIM), TLG(NDIM)
C      CHARACTER*1 REPLY
C      CHARACTER*8 POLARITY
C      LOGICAL G_FLAGX, G_FLAGY, S_FLAG, M_FLAG, ODF_TIME,
ODF_FFT
ALTKM REAL*4 PF_SRCH_BAND, R_ARM, DENSITY, TOTALL, MSAT, MORB,
COMMON/FREQ/DENSITY, TOTALL, MSAT, MORB, ALTKM
COMMON LB, LE, DT, DF, FLOW, FHIGH
C READ IN DATABASE PARAMETERS FROM FILE 'PARAM.DAT'
C
C      PF_SRCH_BAND IS % NUMBER TO COMPUTE SEARCH BAND.
C      R_ARM IS DISTANCE FROM ORBITER C.M. TO CENTER LINE
C      OF DEPLOYER BOOM.
C      ODF_TIME IS LOGICAL FLAG TO CONTROL PRINTING OF TIME
C      DATA TO OUTPUT FILE. THIS FLAG IS NOMINAL .FALSE.
C      MEANING TIME DATA IS NOT PRINTED.
C      ODF_FFT IS LOGICAL FLAG TO CONTROL PRINTING OF FREQUENCY
C      DOMAIN DATA (FFT'S) TO OUTPUT FILE. THIS FLAG IS
C      NOMINAL .TRUE. .. THE DATA IS PRINTED OUT.
C      DENSITY IS TETHER DENSITY IN KG PER KM = 8.35 KG/KM.
C      TOTALL IS TOTAL TETHER LENGTH = 22.0 KILOMETERS.
C      MSAT IS SATELLITE MASS IN KGS. DEFAULT = 510.
C      MORB IS ORBITER MASS IN KGS. DEFAULT = 100,000.
C      ALTKM IS ORBIT ALTITUDE IN KM. DEFAULT = 325. KM.
C
C      OPEN(10,FILE='PARAM.DAT',STATUS='OLD')
C      READ(10,*) PF_SRCH_BAND, R_ARM, ODF_TIME, ODF_FFT
C      READ(10,*) DENSITY, TOTALL, MSAT, MORB, ALTKM
C      CLOSE(10,STATUS='KEEP')
C
C      OPEN FILE COMMANDS
C      THESE ARE OUTPUT FILES FOR RECORD.
C      CLOSE STATEMENTS APPEAR JUST PRIOR TO 'END' STATEMENT.
C
C      OPEN(11,FILE='F_FFTX.DAT',STATUS='UNKNOWN')
C      OPEN(12,FILE='F_FFTY.DAT',STATUS='UNKNOWN')
C      OPEN(13,FILE='F_TXYUV.DAT',STATUS='UNKNOWN')
C      OPEN(17,FILE='F_YAWMAN.DAT',STATUS='UNKNOWN')
C      OPEN(18,FILE='F_RECORD.DAT',STATUS='UNKNOWN')
C
C      THE RATE GYRO DATA SHOULD ALWAYS BE IN LVLH FRAME AND
C      IS THE RATE RELATIVE TO LVLH. I.E. ORBITAL RATE HAS
C      ALREADY BEEN REMOVED. NOTE: PROGRAM WILL STILL WORK

```

```

C      PROPERLY IF ORBITAL RATE IS NOT REMOVED.

C      GO READ FILE FOR TELEMETRY DATA
      CALL READINDATA (TO, TF, X, Y, TLG, JMODE, LF, M_FLAG)
C      CHECK ON M_FLAG STATUS... SET AMCSMODE BY M_FLAG AND JMODE.
C
      PRINT*, 'ESTIMATED TETHER LENGTH IN KILOMETERS IS ', TLG(LF)
      IF (M_FLAG) THEN
        AMCSMODE = JMODE
C      AMCSMODE STATUS ON TELEMETRY NEVER CHANGED FOR ALL
POINTS,
C      OR CHANGED BETWEEN MODES 1 & 2 ONLY.
      ELSE
        PRINT *, ' *****'
        PRINT *, ' WARNING '
        PRINT *, ' *****'
        PRINT *, ' AMCSMODE STATUS CHANGED DURING THIS DATA
STREAM'
        PRINT *, ' NUMBER OF POINTS REDUCED - DATA SET HAS SAME
MODE.'
        PRINT *, ' WILL SET AMCSMODE TO LAST READING'
        AMCSMODE = JMODE
      ENDIF
      PRINT *, ' AMCSMODE IS = ', AMCSMODE
      IF (AMCSMODE.EQ.0.) THEN
        PRINT*

PRINT*, ' *****'
      PRINT*, 'AMCSMODE INDICATES NO VALID DATA - PROGRAM
ABORTS'

PRINT*, ' *****'
      CALL EXIT
      ELSE
        IF (AMCSMODE.EQ.3) THEN
          PRINT*

PRINT*, ' *****'
      PRINT*, 'AMCSMODE INDICATES SPIN CASE - PROGRAM
ABORTS'

PRINT*, ' *****'
      CALL EXIT
      END IF
      END IF

C      SET VALUE FOR FLAG (S_FLAG) TO CONTROL COMPUTATIONS OF YAW
C      MANEUVER OUTPUTS. THIS IS BASED ON AMCSMODE AND TETHER
C      LENGTH.
C      OPERATOR HAS FULL CONTROL OF OPTION TO EXECUTE THIS
COMPUTATION.
C      S_FLAG (LOGICAL) USED TO DENOTE IF YAW MANEUVER

```

```

COMPUTATIONS
C      ARE REQUIRED. IF 'TRUE' MEANS DO THE YAW
COMPUTATIONS.
C      IF 'FALSE' MEANS DO NOT DO THE
COMPUTATIONS.
C
      S_FLAG = .FALSE.
      PRINT*, 'YAW MANEUVER CALCULATIONS WILL NOT BE PERFORMED
UNLESS'
      PRINT*, 'REQUESTED. TYPE YES (Y) TO CALCULATE, NO (N)
OTHERWISE.'
      READ(6,99) REPLY
      IF (REPLY .EQ. 'Y' .OR. REPLY .EQ. 'y') S_FLAG = .TRUE.

C      SAMPLING TIME IS COMPUTED AS AVERAGE VALUE OF ALL DATA
C      THE COMPUTED VALUE CAN BE REPLACED BY OPERATOR -
C      IF OPERATOR SO DESIRES.

C      COMPUTE AVERAGE SAMPLE TIME FOR ALL DATA
C      DT=(LAST TIME - FIRST TIME)/ NUMBER OF POINTS MINUS 1
C      OR READ IN DT.

      2      DT = (TF - T0)/FLOAT(LF-1)
      PRINT *, 'DT IS :', DT, ' IS THIS OKAY TO USE - YES (Y) OR NO
(N)'
      READ (6, 99) REPLY
      IF (REPLY .EQ. 'N' .OR. REPLY .EQ. 'n') THEN
          PRINT *, 'ENTER DT IN SECONDS'
          READ*, DTNEW
      ELSE
          GO TO 4
      ENDIF
C      CHECK TO SEE IF THE NEW DT IS LEGAL
      IR = INT(0.1 + DTNEW/DT)
      IF (IR.LT.2) THEN
          PRINT *, 'CANNOT REDUCE THE DT - MUST USE COMPUTED
VALUE'
          GO TO 4
      ELSE
          I = 1
          J = 1
      3      X(I) = X(J)
          Y(I) = Y(J)
          TLG(I) = TLG(J)
          IF (J+IR .LT. LF) THEN
              I = I + 1
              J = J + IR
              GO TO 3
          ELSE
              LF = I
              GO TO 2
          ENDIF

```



```

PRINT*, 'MAX U = ', UMAX, '    MAX V = ', VMAX
ELSEIF (.NOT.G_FLAGX.AND.G_FLAGX) THEN
  FAVG = FREQX
  TAVG = 1.0 /FAVG
  WK = 1000.*TLNGTH*TAVG/ (360 * PI)
  VMAX = WK * AMPX
  UMAX = 7777.

C
  PRINT *, 'Y AXIS DATA IS SUSPECT - FREQ OUT OF BAND'
  PRINT *, 'FREQUENCY RETURNED IS PREDICTED MIDPOINT'
  PRINT *, 'DATA SHOULD NOT BE USED WITHOUT CAUTION'
  PRINT *, 'MAX V = ', VMAX
  PRINT *, 'NEITHER POLARITY NOR YAW MANEUVER CALCULATIONS ARE
PERFORMED'
  GO TO 88
  ELSEIF (.NOT.G_FLAGX.AND.G_FLAGX) THEN
    FAVG = FREQY
    TAVG = 1.0 /FAVG
    WK = 1000.*TLNGTH*TAVG/ (360*PI)
    UMAX = WK * AMPY
    VMAX = 7777.
    PRINT *, 'X AXIS DATA IS SUSPECT - FREQ OUT OF BAND'
    PRINT *, 'FREQUENCY RETURNED IS PREDICTED MIDPOINT'
    PRINT *, 'DATA SHOULD NOT BE USED WITHOUT CAUTION'
    PRINT *, 'MAX U = ', UMAX
    PRINT *, 'NEITHER POLARITY NOR YAW MANEUVER CALCULATIONS ARE
PERFORMED'
    GO TO 88
    ELSEIF ( (.NOT.G_FLAGX).AND.(.NOT.G_FLAGX) ) THEN
      PRINT *, 'BOTH AXES ARE BAD ***** 3 SUGGESTIONS'
      PRINT *, '1) SUGGEST LOOK AT TIME PLOTS OF GYRO
SIGNALS.'
      PRINT *, '2) SUGGEST MAKE FFT PLOTS AND LOOK AT DATA.'
      PRINT *, '3) SUGGEST WIDENING SEARCH BAND.'
      UMAX = 7777.
      VMAX = 7777.
      FAVG = 7777.
      TAVG = 7777.
      WK = 7777.
      PSIGN= 7777.
      GO TO 88

C
C
C
  DATA IS BAD. WRITE OUTPUT AT LABEL '88'

  ENDIF

  PRINT *, 'AVERAGE PERIOD IN SECONDS IS :',TAVG

C
C
C
C
  THE INTEGERS 'IWXMAX' AND 'IWYMAX' ARE TIME INDICES WHERE X
  AND Y VALUES ARE A MAXIMUM. THIS CORRESPONDS TO WHERE
  COSINE(PHI) = 1 OR PHI = 2*PI.
  IF IWYMAX GT IWXMAX , MEANS POLARITY IS POSITIVE ABOUT Z

```

```

C      PROVIDED THE TIME DIFFERENCE BETWEEN IWYMAX AND IWXMAX IS
C      EQUIVALENT TO 90 DEGREES.
C      IWYMAX COULD BE GREATER THAN IWXMAX FOR NEGATIVE ROTATION
C      BUT THIS WOULD REQUIRE A 270 DEGREE TRAVEL TIME.
C      THUS THE TEST FOR POLARITY IS 180 DEGREES TRAVEL TIME.
C
C      IF (IWYMAX .GT. IWXMAX) THEN
C          TEST = DT*FLOAT(IWYMAX-IWXMAX)
C          TEST IS TIME IN SECONDS TO GO FROM X-AXIS TO Y-AXIS.
C          POLARITY DICTATED BY THIS TIME BEING GT OR LT 1/2 OF
PERIOD
C
C          IF (TEST .GT. 0.5*TAVG) THEN
C              PSIGN = -1.0
C          ELSE
C              PSIGN = +1.0
C          ENDIF
C      ENDIF
C
C      NOW DO CASE FOR X PEAK OCCURS AFTER Y PEAK
C      THIS IS SAME LOGIC AS ABOVE IN PRINCIPLE
C
C      IF (IWXMAX .GT. IWYMAX) THEN
C          TEST = DT*FLOAT(IWXMAX - IWYMAX)
C          IF (TEST .GT. 0.5*TAVG) THEN
C              PSIGN = +1.0
C          ELSE
C              PSIGN = -1.0
C          ENDIF
C      ENDIF
C
C      ALL DONE - SIGN COMPUTATIONS ARE COMPLETED
C      PRINT *, ' POLARITY OF SKIPROPE = ', PSIGN
C      PRINT*
C
C      WRITE TIME DATA TO FILE FOR RECORD ONLY IF REQUESTED.
C      REQUEST IS IF ODF_TIME FLAG IS TRUE.
C
C      IF (ODF_TIME) THEN
C          DO I = 1, LEB
C              TWX = COS(2.0*PI*FAVG*DT*(I-1)+PHASEX)
C              TWY = COS(2.0*PI*FAVG*DT*(I-1)+PHASEY)
C              WX = TWX * AMPX
C              WY = TWY * AMPY
C              U = -PSIGN * WK * AMPY * TWX
C              V = -PSIGN * WK * AMPX * TWY
C              T = T0 + (LB + I - 2)*DT
C              WRITE(13,*)T,WX,WY,U,V
C          END DO
C      ENDIF
C
C      IF STATION 2 FLAG SET TO 'TRUE', THEN DO YAW MANEUVER

```

```

C      CALCULATIONS, OTHERWISE SKIP TO LABEL 88.
C
C      IF (S_FLAG) THEN
C
C          CALCULATE NUMBER OF ROTATIONS ORBITER SHOULD EXECUTE.
C
C          RNROT = ( AMIN1 (UMAX, VMAX) ) / (2.0 * R_ARM)
C
C          SPECIFY POLARITY
C          IF (PSIGN .EQ. 1.) THEN
C              POLARITY = 'POSITIVE'
C          ELSE
C              POLARITY = 'NEGATIVE'
C          ENDIF
C
C          OUTPUT REV NUMBER, POLARITY, # OF ROTATIONS, AND START
TIMES
C          THIS DATA ALSO GOES TO FILE. DATA SET TIME TAG IS GIVEN
BY 'TMIDPT'.
C
C          BURN TIMES CALCULATED HERE ASSUME THAT THE ORBITER NOSE IS
C          ALIGNED WITH THE X-LVLH AXIS.....
C          IF THIS IS NOT THE CASE, THE BURN TIMES MUST BE ADJUSTED TO
C          ACCOUNT FOR WHERE THE NOSE IS WRT THE X LVLH AXIS.....
C          THIS TIME ADJUSTMENT IS : (YAW ANGLE/360*FAVG) IN SECONDS.
C          YAW ANGLE IS DEGREES NOSE IS AWAY FROM X-LVLH.
C          FAVG IS VALUE FROM FREQUENCY MEASUREMENTS (IN HZ).
C
C
C          CALCULATE 5 BURN TIMES AND OUTPUT TO SCREEN AND FILE 17
C          FIRST, ESTABLISH TIME WHEN X AND Y ARE MAXIMUMS - TXMAX
& TYMAX
C
C          TXMAX = DT*(IWXMAX-1)
C          TYMAX = TXMAX + 0.25*TAVG + TSHIFT + TO
C          WRITE (6,76) TMIDPT, TF
C          WRITE (6,*)
C          WRITE (17,76) TMIDPT, TF
C          WRITE (17,77)
76      FORMAT (' MIDPOINT TIME FOR THIS DATA WINDOW IS
: ',E18.6, /
$          , ' TIME TAG ON LAST POINT IN BUFFER IS : ',E18.6)
C          PRINT*
77      FORMAT(' REV LABEL POLARITY RATE(D/S) # OF
ROTATIONS '
$          , ' START TIME' )
C
C          NOW ADJUST BURN START TIMES TO ACCOUNT FOR ORBITER
ORIENTATION.
C          THIS REQUIRES OPERATOR INPUT FOR ORBITER YAW ANGLE.
C
61      PRINT*, 'ENTER ORBITER YAW AXIS ANGLE IN DEGREES'

```

```

      READ *, OYAWANG

      PRINT*, 'ORBITER NOSE IS ', OYAWANG, ' DEGREES WRT X-LVLH
AXIS'
      PRINT*, 'IS THIS CORRECT (Y OR N)?'
      READ(6,99) REPLY

      IF (REPLY .EQ. 'N' .OR. REPLY .EQ. 'n') GO TO 61

      BTDEL = PSIGN * OYAWANG/(360.0*FAVG)
      TYMAX = TYMAX + BTDEL
      WRITE(6,77)

C
C   ADVANCE TIME 'TYMAX' BY INCREMENTS OF TAVG UNTIL 'TIME'
COMES
C   UP THAT IS GREATER THAN TIME OF LAST DATA POINT IN BUFFER
(TF).
C   ANY COMPUTED TIMES LESS THAN 'TF' ARE IN THE PAST.
C
      78 IF (TYMAX .LE. TF) THEN
          TYMAX = TYMAX + TAVG
          GO TO 78
      ENDIF

C
C   NOW COMPUTE THE 5 BURN TIMES
C
      DO K=1, 5
          TT = TYMAX + (K-1)*TAVG
          WRITE(6,17) K-1, POLARITY, 360.*FAVG, RNROT, TT
          WRITE(17,17) K-1, POLARITY, 360.*FAVG, RNROT, TT
      END DO
      17   FORMAT(I5,7X,A8,F12.4,7X,F5.1,7X,F12.2)

      ENDIF

      88 CONTINUE

      CALL ODTF (T0, TMIDPT, TF, DT, LE, LB, LEB, TLNGTH
$      , AMPX, FREQX, PHASEX, AMPY, FREQY, PHASEY
$      , FAVG, TAVG, WK, PSIGN, UMAX, VMAX, FLOW, FHIGH
$      , AVGX, AVGY)

C
C
C
C
C   CLOSE ALL FILES

      CLOSE (11, STATUS='KEEP')
      CLOSE (12, STATUS='KEEP')
      CLOSE (13, STATUS='KEEP')
      CLOSE (17, STATUS='KEEP')
      CLOSE (18, STATUS='KEEP')
      END

```

```

C      SUBROUTINE WORK CALCULATES THE AMPLITUDE, PHASE, AND
FREQUENCY
C      OF THE DATA.  THE FOURIER TRANSFORM SUBROUTINE FOUR1 IS
C      CALLED BY SUBROUTINE WORK.  WORK RETURNS TO THE MAIN
PROGRAM
C      THE VALUES OF THE AMPLITUDE, PHASE, AND FREQUENCY AS WELL AS
C      THE TIME INDEX WHERE THE MAXIMUM VALUE OCCURS.
C      THIS IS BASED ON MODEL OF COS(WT+PHASE).
C
      SUBROUTINE WORK (IOA, ANG, AMP, PHASE, FREQ, ITMAX
$, G_FLAG, BIAS, FFT_FLAG )

      INTEGER NDIM, NCDIM
      PARAMETER (NDIM=3000, NCDIM=8200, NFT=8192, NPNT=7)
      PARAMETER (PI=3.1415926, DFR=57.2957795, RFD=0.017453292)
      DIMENSION AUX(NDIM), ANG(1)

      REAL*4 XFREQ(7), PHIMAG(7), PHREAL(7)
      COMPLEX AWO(NCDIM)
      LOGICAL G_FLAG, FFT_FLAG

      COMMON LB, LE, DT, DF, FLOW, FHIGH

      NTB1=LE-LB+1
      NTB1 IS FORCED TO BE ODD IN MAIN PROGRAM.
      HANN WINDOW ROUTINE USES ODD NUMBER OF POINTS TO TAPER.

      LOAD INPUT DATA FROM ANG(I) INTO ARRAY AUX(J).
      LB1=1-LB
      DO I=LB, LE
        IL=I+LB1
        AUX(IL)=ANG(I)
      END DO

      APPLY WINDOW FUNCTION TO TIME SEQUENCE
      CALL HANN (NTB1, AUX, BIAS)

      MAKE COMPLEX NUMBER AWO(I) FROM REAL NUMBER AUX(I) BY USING
      A ZERO IMAGINARY VALUE (AUX(I) IS THE REAL VALUE).

      DO I=1, NTB1
        AWO(I)=CMPLX(AUX(I), 0.)
      END DO

      NOW PAD THE DATA STREAM WITH ZEROS OUT TO AWO(8192).

```

```

DO I=NTB1+1,NFT
    AWO(I) = CMPLX(0.,0.)
END DO

C
C    SUBROUTINE FOUR1 DOES THE FOURIER TRANSFORM USING A FFT
METHOD.
C
    CALL FOUR1(AWO,NFT,1)

C    NOW FIND THE MAXIMUM MODULUS VALUE OF THE FOURIER TRANSFORM
DATA
C    OVER A SPECIFIED FREQUENCY INTERVAL. THIS INTERVAL IS
CALCULATED
C    FROM INPUT DATA AND IS DESIGNED SUCH THAT THE SKIPROPE
C    FREQUENCY FALLS WITHIN THIS INTERVAL. THE INTERVAL IS
C    SUFFICIENTLY NARROW THAT NO OTHER MODE SHOULD FALL WITHIN
THE
C    INTERVAL.

C    FLOW IS LOWER BOUNDARY OF SEARCH BAND
C    FHIGH IS UPPER BOUNDARY OF SEARCH BAND
C
    XMAX=0.0
    IFRST = 1 + INT( FLOW/DF)
    ILAST = 1 + INT( FHIGH/DF)
    DO I = IFRST, ILAST
        FR = (I-1)*DF
        IF(CABS(AWO(I)).GT.XMAX) THEN
            XMAX=CABS(AWO(I))
            KF=I
            FREQ = FR
        END IF
    END DO

C    CREATE THE 3 DATA SETS TO BE FITTED BY LEAST SQUARES
POLYNOMIAL.
C    POLYNOMIAL IS 2ND DEGREE AND 7 POINTS WILL BE USED IN CURVE
FIT.
C        3 SETS ARE:
C            MAGNITUDE OF TRANSFORM (SQRT(REAL**2 + IMAG**2))
C            REAL PART
C            IMAGINARY PART
C
C    CENTER OF DATA SET IS THE FREQUENCY POINT WHERE MAX WAS
FOUND.
C
    DO I = 1,NPNT
        J = KF-((NPNT+1)/2.0)+I
        XFREQ(I) = CABS(AWO(J))
    
```



```

        PHIMAG(I) = AIMAG(AWO(J))
        PHREAL(I) = REAL(AWO(J))
    END DO

C
C
C    DO CURVE FIT ON THE MODULUS OF THE FOURIER TRANSFORM
C    CALL TO LSCF WITH OPTION 1 DOES 2 THINGS.
C    CURVE FITS AND COMPUTES TRUE MAX FREQUENCY POINT.
C
C    CALL LSCF (FQ_PO, XMAX, XFREQ, 1, G_FLAG)
C
C    G_FLAG = TRUE MEANS MAX FREQUENCY FOUND IN THE SPECIFIED
INTERVAL.
C    LSCF IS THEN CALLED TWICE WITH OPTION 2 TO EVALUATE THE
POLYNOMIAL
C    AT THE CRITICAL FREQUENCY VALUE FOUND IN THE FIRST LSCF
CALL.
C    IF FALSE, THEN THE MAXIMUM PEAK OCCURS OUTSIDE THE 7 POINT
RANGE.
C    (THIS IS POSSIBLE IF THE INITIAL ESTIMATE OF THE SKIPROPE
FREQUENCY
C    ESTIMATED FROM THE TETHER LENGTH IS MUCH DIFFERENT FROM THE
TRUE VALUE.) WHEN G_FLAG IS FALSE THE INFORMATION RETURNED
ARE
C    VALUES BASED ON THE MIDPOINT OF THE SPECIFIED SEARCH BAND.
C
C    IF (G_FLAG) THEN
CALL LSCF (FQ_PO, PHASEI, PHIMAG, 2, G_FLAG)
CALL LSCF (FQ_PO, PHASER, PHREAL, 2, G_FLAG)
ELSE
    KF = 1+ INT(((FLOW+FHIGH)/2.0)/DF)
    XMAX = CABS(AWO(KF))
    PHASEI = AIMAG(AWO(KF))
    PHASER = REAL(AWO(KF))
    FREQ = 0.
    FQ_PO = KF - 1.0
END IF
FREQ = FREQ + DF * FQ_PO

C
C    SCALING OF TRANSFORMED DATA IS PERFORMED TO GIVE OUTPUTS IN
C    DEGS/SEC AND REPRESENT ACTUAL RATE DATA.
C
SCALE = 4.0/FLOAT(NTB1-1)
AMP = SCALE * XMAX
PHASE = -ATAN2(PHASEI, PHASER)

C
C    THE FORWARD FOURIER TRANSFORM IS USUALLY DEFINED WITH
EXP(-i*PI*F*T).
C    MANY FFT ROUTINES, INCLUDING FOUR1, USE EXP(+i*2*PI*F*T).
THESE TWO
C    DIFFERENT CONVENTIONS FOR THE FORWARD FOURIER TRANSFORM
RESULT IN TWO

```

```

C      DIFFERENT FORMS FOR THE SHIFT THEOREM.  IN THE FIRST CASE,
THE SHIFT
C      THEOREM STATES THAT IF G(T) TRANSFORMS AS G(F), THEN
G(T+T1) TRANSFORMS
C      AS  $\exp(i*2*PI*F*T1)*G(F)$ .  IN THE SECOND CASE, IF G(T)
TRANSFORMS AS
C      G(F), THEN G(T+T1) TRANSFORMS AS  $\exp(-i*2*PI*F*T1)*G(F)$ .
SINCE OUR
C      MODEL IS  $\cos(2*PI*F*T + P) = \cos(2*PI*F*(T + P/(2*PI*F)))$ ,
AND WE USE
C      THE FIRST CONVENTION FOR THE FOURIER TRANSFORM, WE EXPECT
OUR PHASE
C      TO BE  $2*PI*F*P/(2*PI*F) = P$ .  HOWEVER, SINCE THE PROGRAM
USES THE
C      SECOND CONVENTION FOR THE FOURIER TRANSFORM, THE PHASE IS
-P, SO TO
C      CORRECT FOR THIS DIFFERENCE WE MUST INCLUDE ANOTHER - SIGN:
-(-P) = P.

C      CALCULATE THE TIME INDEX WHERE THE MAXIMUM RATE OCCURS.
C      THIS IS BASED ON  $\cos(\phi) = 1$  IMPLIES  $\phi = 2*PI$ 
C
ITMAX=INT((1.0/(FREQ*DT))*(1.0-PHASE/(2.0*PI))+0.5)+1

C
C      IF THIS INDEX CORRESPONDS TO A TIME GREATER THAN 1 PERIOD,
C      THEN SUBTRACT THE EQUIVALENT OF 1 PERIOD FROM ITMAX.
C
IF (ITMAX*DT .GT. (1./FREQ) ) THEN
    ITMAX = ITMAX - INT ( 1.0/(FREQ*DT) )
ENDIF

C
C      OUTPUT THE MODULUS OF THE TRANSFORM FROM 0.0 HZ THROUGH THE
PENDULOUS
C      FREQUENCY (ASSUMING MAX VALUE IS 0.04 HZ), USING A SPACING
OF DF.
C      OUTPUT 4 NUMBERS PER LINE: FREQ(HZ), MODULUS(DEG/SEC), REAL
PART,
C      AND IMAGINARY PART. (NOTE: LAST TWO ARE NOT IN DEG/SEC)
C       $DF=1./(8192*8*0.128) = 1./8388.6 = 0.0001192$ 
C       $KQ1 = .04/DF = 335.54$  --- CALL THIS 336
C
C      WRITE FFT DATA TO OUTPUT FILE IF FFT_FLAG IS TRUE
C      OTHERWISE DO NOT WRITE TO OUTPUT.
C
IF (FFT_FLAG) THEN
    ILAST = 1006
    IF (FREQ .GT. .0035 ) ILAST = 336
    DO I = 1, ILAST
        FR = (I-1)*DF
        XMAX = SCALE*CABS(AWO(I))
        WRITE (IOA,*) FR, XMAX, REAL(AWO(I)), AIMAG(AWO(I))
    END DO

```

```

ENDIF
RETURN
END

```

```

SUBROUTINE HANN (LA,A11, BIAS)

```

```

C
C TAPER IS RAISED COSINE CURVE.
C MEAN IS COMPUTED AND REMOVED FROM INPUT SIGNAL
C PARAMETER (PI=3.1415926)
REAL A11(LA), HW(3000)
ITM = (LA-1)/2
RM = FLOAT(ITM)
DO IT= -ITM, ITM
  I = 1 + IT + ITM
  HW(I)=0.5*(1.0 + COS(PI*FLOAT(IT)/RM ) )
  A11(I)=A11(I)*HW(I)
END DO
C COMPUTE MEAN OF TAPERED SIGNAL
C TRUE MEAN IS TWICE COMPUTED VALUE BECAUSE HANN WINDOW
C REDUCES VALUE BY FACTOR OF 2. (I.E. MEAN OF WINDOW IS 0.5)
C
CALL MEAN (LA, A11, BIAS)
BIAS = BIAS * 2.0 * LA/(LA-1)
DO I = 1,LA
  A11(I) = A11(I) - BIAS * HW(I)
END DO
RETURN
END

```

```

SUBROUTINE MEAN(LA,A22, SA)

```

```

C
C THIS ROUTINE COMPUTES THE DC TERM OF THE DATA STREAM.
C MEAN IS NOT REMOVED, BUT ONLY COMPUTED.
C
REAL A22(LA)
SA = 0.
DO I=1,LA
  SA=SA+A22(I)
END DO
SA=SA/FLOAT(LA)
RETURN
END

```

```

SUBROUTINE FOUR1(DATA,NN,ISIGN)

```

```

C
C THIS ROUTINE DOES THE FOURIER TRANSFORM USING A FFT METHOD.
C
REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
DIMENSION DATA(*)

```

```

N=2*NN
J=1
DO 11 I=1,N,2
  IF (J.GT.I) THEN
    TEMPR=DATA(J)
    TEMPI=DATA(J+1)
    DATA(J)=DATA(I)
    DATA(J+1)=DATA(I+1)
    DATA(I)=TEMPR
    DATA(I+1)=TEMPI
  ENDIF
  M=N/2
1  IF ((M.GE.2).AND.(J.GT.M)) THEN
    J=J-M
    M=M/2
    GO TO 1
  ENDIF
  J=J+M
11 CONTINUE
  MMAX=2
2  IF (N.GT.MMAX) THEN
    ISTEP=2*MMAX
    THETA=6.28318530717959D0/(ISIGN*MMAX)
    WPR=-2.D0*DSIN(0.5D0*THETA)**2
    WPI=DSIN(THETA)
    WR=1.D0
    WI=0.D0
    DO 13 M=1,MMAX,2
      DO 12 I=M,N,ISTEP
        J=I+MMAX
        TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
        TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
        DATA(J)=DATA(I)-TEMPR
        DATA(J+1)=DATA(I+1)-TEMPI
        DATA(I)=DATA(I)+TEMPR
        DATA(I+1)=DATA(I+1)+TEMPI
12      CONTINUE
        WTEMP=WR
        WR=WR*WPR-WI*WPI+WR
        WI=WI*WPR+WTEMP*WPI+WI
13      CONTINUE
        MMAX=ISTEP
      GO TO 2
    ENDIF
    RETURN
  END

```

C THIS SUBROUTINE DOES LEAST SQUARES CURVE FIT TO 7 POINTS
 C FOR A 2ND DEGREE POLYNOMIAL. THE DATA IS ASSUMED TO BE
 C SAMPLED
 C AT INTEGRAL INTERVALS. ANY SCALING MUST BE DONE OUTSIDE

```

C      THIS SUBROUTINE.  THE 7 POINTS ARE :
C      P = -3, -2, -1, 0, 1, 2, 3
C      THE POLYNOMIAL IS  $F(P) = A + B \cdot P + C \cdot P^2$ .
C      THE MAX OCCURS AT  $P = P_0 = -B/(2 \cdot C)$ .
C      FREQUENCY CORRESPONDING TO  $P_0$  IS  $P_0 \cdot DF$  (DF OF DATA
STREAM)
C      THIS DELTA IS REFERENCED TO MIDPOINT FREQUENCY OF 7
POINTS.
C      THE MAX VALUE IS  $F(P_0) = A - (B^2)/(4 \cdot C)$ .
C
C      SUBROUTINE LSCF (P0, FMAX, U_IN, IOPT, G_FLAG)
C
C      ON ENTRY:
C      U_IN IS INPUT ORDINATE VALUES. (7 )
C      IOPT IS OPTION FOR 1 OF 2 THINGS
C      1 : FIND P0 WHERE MAX OCCURS PLUS COMPUTE MAX VALUE.
C      2 : COMPUTE VALUE OF POLYNOMIAL AT SPECIFIED FREQUENCY
P0.
C      IF IOPT=2, THEN P0 IS FREQUENCY POINT TO EVALUATE
POLYNOMIAL.
C
C      ON EXIT
C      P0 IS VALUE OF P WHERE MAX PEAK OCCURS;
C      THIS IS WRT CENTER POINT OF DATA.
C      FMAX IS VALUE OF FUNCTION AT  $P=P_0$ .
C      G_FLAG IS FLAG FOR DATA VALIDITY
C      SET TO TRUE IF EVERYTHING IS OKAY
C      SET TO FALSE IF PEAK IS OUTSIDE SEARCH ZONE
C      *****
C      REAL*4 U_IN(*)
C      LOGICAL G_FLAG
C      *****
C      FIRST STEP IS TO DO LEAST SQUARES.
C      ALL COEFFICIENTS HAVE BEEN PRE-COMPUTED.
C      'A' IS -8, 12, 24, 28, 24, 12, -8 DIVIDED BY 84
C      'B' IS -9, -6, -3, 0, 3, 6, 9 DIVIDED BY 84
C      'C' IS 5, 0, -3, -4, -3, 0, 5 DIVIDED BY 84
C
C      US17 = U_IN(1) + U_IN(7)
C      US35 = U_IN(3) + U_IN(5)
C      A = COF1
C      COF1 = -8.*US17 + 12.*(U_IN(2)+U_IN(6)) +
#          24.*US35 + 28.*U_IN(4)
C      B = COF2
C      COF2 = 9.*(-U_IN(1)+U_IN(7) ) +
#          6.*(-U_IN(2)+U_IN(6) ) +
#          3.*(-U_IN(3)+U_IN(5) )
C      C = COF3
C      COF3 = 5.*US17 - 3.*US35 - 4.*U_IN(4)
C      IF (ABS(COF3).LT.1.0E-08) THEN
C          PRINT*, '***** WARNING *****'
C          PRINT*, 'CONSTANT VALUE EQUALS ZERO - CANNOT COMPUTE A

```

```

MAXIMUM'
      PRINT*, 'FREQUENCY VALUE.'
      G_FLAG = .FALSE.
      RETURN
    ENDIF

C
C   DID NOT DIVIDE BY 84 YET. DO SO FOR MAX PART BUT NOT P0.
C
C   COMPUTE P0, VALUE OF F(P) WHERE  $A+B*P+C*P*P = 0$ 
C   COMPUTE FUNCTION AT P0;  $A+B*P0+C*P0*P0 = A-B**2/4C$ 
C   IF (IOPT .EQ. 1) THEN
      P0=-0.5*COF2/COF3
      FMAX = (COF1 + 0.5 * P0 * COF2) /84.
      G_FLAG = .TRUE.

      IF (ABS(P0) .GT. 3.0) THEN
        PRINT*, '* * * WARNING * * *'
        PRINT*, 'HAVE NO MAX VALUE IN SPECIFIED INTERVAL'
        G_FLAG = .FALSE.
      ENDIF
    ELSE
      FMAX = (COF1 + P0*( COF2 + P0*COF3 ) )/84.
    END IF
    RETURN
  END

  SUBROUTINE FBAND (FL, FH, TLKM, PF)

C
C   SUBROUTINE RETURNS FL AND FH; THESE ARE LOW AND HIGH
C   FREQUENCY
C   VALUES TO SEARCH FOR PEAK AMPLITUDE.
C   THIS SUBROUTINE COMPUTES SKIPROPE FREQUENCY (FC) FROM THE
C   MASSES
C   OF THE ORBITER AND SATELLITE AND THE TETHER LENGTH. DATA
C   NEEDED
C   FOR THESES CALCULATIONS ARE IN COMMON BLOCK 'FREQ' AND ARE
C   READ
C   FROM FILE 'PARAM.DAT' IN MAIN PROGRAM.
C   TKLM IS THE TETHER LENGTH IN KILOMETERS.
C   PF IS PERCENT OF SKIPROPE FREQUENCY TO USE AS A DELTA 'F',
C   I.E.,
C   BAND IS FROM FC - DELTA (FL) TO FC + DELTA (FH).
C
  REAL*4 DENSITY, TOTALL, MSAT, MORB, ALTKM, FL, FH, FC,
  TLKM, PF
  COMMON/FREQ/DENSITY, TOTALL, MSAT, MORB, ALTKM
  OMSQ = ORBRATESQ (ALTKM)
  CK = 3.0 * OMSQ / DENSITY
  MO = MORB + TOTALL * DENSITY
  Q = 0.5 * DENSITY * TLKM
  MSTAR = ((MO-Q)*(MSAT+Q)) / (MO+MSAT)

```

FC = 0.5 * SQRT(CK*MSTAR/TLKM)

DF = FC*PF/100.

FL = FC - DF

FH = FC + DF

RETURN

END

REAL FUNCTION ORBRATESQ (ALTKM)

PARAMETER (GM = 9.81098, RE = 6378.17)

R = GM/(1000.0*RE)

ORBRATESQ = R/(1.0 + ALTKM/RE)**3

RETURN

END

SUBROUTINE READINDATA (TO, TF, X, Y, TLG, MODE, LF, MFLAG)

SUBROUTINE READS IN DATA FROM FILE. THIS IS EQUIVALENT
TO DATA THAT WILL COME FROM THE PREPROCESSOR BLOCK.

REAL*4 X(1), Y(1), TLG(1)

INTEGER MODE

LOGICAL MFLAG

TO : TIME TAG FOR 1ST POINT IN BUFFER

TF : TIME TAG FOR LAST POINT IN BUFFER

X(I) : X AXIS GYRO DATA (DEG/SEC)

Y(I) : Y AXIS GYRO DATA (DEG/SEC)

TLG(I) : TETHER LENGTH IN KILOMETERS

MODE : AMCSMODE VALUE FOR LAST TIME POINT 'TF'

AMCSMODE = 0, 1, 2, OR 3

AMCSMODE = 0 - NO VALID DATA

AMCSMODE = 1 - PASSIVE CASE

AMCSMODE = 2 - YAW HOLD

AMCSMODE = 3 - SPIN CASE

LF : NUMBER OF TIME POINTS STORED IN X & Y ARRAYS

MFLAG : LOGICAL TO DENOTE IF AMCSMODE CHANGED FROM 1 OR

TO A 0 OR 3 BETWEEN TIMES OF TO TO TF.

UNOMSC.FOR READS FROM FORTRAN FILE 10, 5 NUMBERS PER LINE.

TIME, X-GYRO RATE, Y GYRO RATE, TETHER LNTH, AMCSMODE

TIME DATA SHOULD BE IN SECONDS.

GYRO RATE DATA SHOULD BE IN DEGS/SEC.

TETHER LENGTH IN KILOMETERS.

MODE IS INTEGER BETWEEN 0 AND 3.

MFLAG = .TRUE.

MFLAG IS LOGICAL VARIABLE TO DENOTE THAT MODE STATUS IS

C CONSISTENT FOR THE FULL DATA STREAM. IF MODE CHANGES TO
 C 0 (NO VALID DATA) OR 3 (SPIN), THEN THE READ OPERATION IS
 C STOPPED AND THE BUFFER WILL HAVE A REDUCED NUMBER OF
 POINTS.
 C THE OPERATOR IS NOTIFIED BY A MESSAGE PRINTED TO THE
 SCREEN.

C
 C OPEN(10, FILE='IDFTXY.DAT', STATUS='UNKNOWN')
 C READ FIRST LINE TO GET FIRST TIME

READ (10, *, END=2) TO, X(1), Y(1), TLG(1), JMODE

C LOOP TO READ DATA

I=2

1 READ(10, *, END=2) TF, X(I), Y(I), TLG(I), MODE

I = I + 1

IF (JMODE .EQ. MODE) THEN

GO TO 1

ELSE

IF ((MODE*JMODE) .EQ. 2) THEN

JMODE = MODE

GO TO 1

ELSE

MFLAG = .FALSE.

GO TO 2

ENDIF

ENDIF

2 LF=I-1

CLOSE(10, STATUS='KEEP')

C
 C RETURN
 C END

SUBROUTINE ODTF (TO, TM, TF, DT, LE, LB, LEB, TL
 \$, AX, FX, PX, AY, FY, PY
 \$, FA, TA, WK, PSIGN, UMAX, VMAX, FL, FH
 \$, AVGX, AVGY)

C
 C WRITE DATA IN ARGUMENT TO FILE

C
 C DEFINITION OF DATA ELEMENTS

C TO : 1ST TIME POINT IN BUFFER (MET)

C TM : TIME AT MIDPOINT OF DATA WINDOW (MET)

C TF : LAST TIME POINT IN BUFFER (MET)

C DT : SAMPLE TIME - SECONDS

C LE : INDEX OF LAST POINT IN DATA WINDOW

C LB : INDEX OF FIRST POINT IN DATA WINDOW

C LEB : NUMBER OF POINTS USED IN DATA WINDOW

C TL : TETHER LENGTH FOR THIS CASE - KILOMETERS

C AX : PEAK MAGNITUDE OF RATE IN X AXIS - DEG/SEC


```

C      FX      : MEASURED FREQUENCY OF SKIPROPE IN X AXIS - HZ
C      PX      : PHASE ANGLE IN X AXIS
C      AY      : PEAK MAGNITUDE OF RATE IN Y AXIS - DEG/SEC
C      FY      : MEASURED FREQUENCY OF SKIPROPE IN Y AXIS - HZ
C      PY      : PHASE ANGLE IN Y AXIS
C      FA      : AVERAGE FREQUENCY OF 'GOOD AXES'
C      TA      : AVERAGE PERIOD OF 'GOOD AXES'
C      WK      : CONVERSION CONSTANT FROM RATE TO U ; METER-SEC/DEG
C      PSIGN   : DIRECTION OF SKIPROPE - WRT Z LVLH AXIS
C      UMAX    : MAXIMUM VALUE OF MIDNODE IN X DIRECTION
C      VMAX    : MAXIMUM VALUE OF MIDNODE IN Y DIRECTION
C      FL      : LOWER BOUNDARY OF FREQUENCY SEARCH BAND - HZ
C      FH      : UPPER BOUNDARY OF FREQUENCY SEARCH BAND - HZ
C      AVGX    : COMPUTED MEAN OF X AXIS TIME SEQUENCE
C      AVGY    : COMPUTED MEAN OF Y AXIS TIME SEQUENCE
C

```

```

WRITE(18,*) TM
WRITE (18,*) LB, LE, LEB
WRITE (18,*) TO, TF, DT, TL, WK, PSIGN
WRITE(18,*) AX, FX, PX, AY, FY, PY
WRITE(18,*) FA, TA, UMAX, VMAX, FL, FH
WRITE(18,*) AVGX, AVGY

```

C

```

RETURN
END

```

APPENDIX 2

APPENDIX 2.A

Model Signal Generation Programs with Random Number Generator and Box-Muller Algorithm for Gaussian Distributed Variates

APPENDIX 2.B

Results of ECR Verification Table 2. Filter Test Cases.

APPENDIX 2.C

Programs for ECR Testing

- (1) SIMREAD - Read Simulation Data Files
- (2) SIMTEST - Compare Observer Output to Simulation Input
- (3) BTBUSO - Compare Observer Output to Model Signal Input

APPENDIX 2.D

Programs for Systematic Testing

- (1) NO_NOISE - Noise-free Test Cases, Station 2
- (2) NOISE - Noisy Test Cases, Station 2
- (3) LIBRATION - Noise-free Tests at Station 1
- (4) LIB_NOISE - Noisy Tests at Station 1

APPENDIX 2.E

Simulation Test Results

APPENDIX 2.F

Systematic Test Results

- (1) NO_NOISE - Noise-free Test Cases, Station 2
- (2) NOISE - Noisy Test Cases, Station 2
- (3) LIBRATION - Noise-free Tests at Station 1
- (4) LIB_NOISE - Noisy Tests at Station 1

APPENDIX 2.A

Model Signal Generation Programs with Random Number Generator and
Box-Muller Algorithm for Gaussian Distributed Variates

The following random number generator is documented on page 196 of "Numerical Recipes" (Press, Flannery, Teukolsky, and Vetterling, 1989, Cambridge University Press). Setting the initial value idum to a negative integer initializes the routine.

```
function ran1(idum)
dimension r(97)
parameter (m1 = 259200, ia1 = 7141, ic1 = 54773, rm1 = 1.0/m1)
parameter (m2 = 134456, ia2 = 8121, ic2 = 28411, rm2 = 1.0/m2)
parameter (m3 = 243000, ia3 = 4561, ic3 = 51349)
data iff /0/
if (idum.lt.0.or.iff.eq.0) then
    iff = 1
    ix1 = mod(ic1 - idum,m1)
    ix1 = mod(ia1*ix1 + ic1,m1)
    ix2 = mod(ix1,m2)
    ix1 = mod(ia1*ix1 + ic1,m1)
    ix3 = mod(ix1,m3)
    do j = 1,97
        ix1 = mod(ia1*ix1 + ic1,m1)
        ix2 = mod(ia2*ix2 + ic2,m2)
        r(j) = (float(ix1) + float(ix2)*rm2)*rm1
    end do
    idum = 1
end if
ix1 = mod(ia1*ix1 + ic1,m1)
ix2 = mod(ia2*ix2 + ic2,m2)
ix3 = mod(ia3*ix3 + ic3,m3)
j = 1 + (97*ix3)/m3
if (j.gt.97.or.j.lt.1) pause
ran1 = r(j)
r(j) = (float(ix1) + float(ix2)*rm2)*rm1
return
end
```

The following Box-Muller algorithm for generating normally (Gaussian) distributed variates is documented on page 202 of "Numerical Recipes". The random number generator ran1 listed above is used in this algorithm.

```
function gasdev(idum)
data iset/0/
if (iset.eq.0) then
    v1 = 2.0*ran1(idum) - 1.0
    v2 = 2.0*ran1(idum) - 1.0
    r = v1**2 + v2**2
    if (r.ge.1) go to 1
```

```

C Program CREATE.FOR generates model data sets for the UNOMSC.FOR backup skip-
C rope observer program. Five numbers are outputted per line of the data file
C IDFTXY.DAT: time, x (roll) axis gyro rate, y (pitch) axis gyro rate, tether
C length, and amcsmode. The user can opt to include noise and/or dropouts into
C the data. If desired, the program will estimate the skiprope and pendulous
C frequencies as functions of the tether length. (Note: These estimated fre-
C quencies are exactly that - estimates - and definitely should not be con-
C strued as accurate values!)
C     The operator is prompted to input the following values:
C         time sampling interval, beginning time tag
C         number of data points, standard deviation, and random seed
C         (standard deviation is 10 X the nominal value of 2.8e-04)
C         (random seed should be a negative seed)
C         skiprope amplitude, frequency, and phase (for x axis)
C         skiprope amplitude, frequency, and phase (for y axis)
C         1 or 2 to add noise or not, dropout percentage (0.0 to 1.0)
C         orb rate, pendulous amplitude, frequency, and phase (for x axis)
C         orb rate, pendulous amplitude, frequency, and phase (for y axis)
C         tether length, amcsmode
C         0 to use inputted values to generate model, 1 to estimate skiprope
C         and pendulous frequencies from the inputted tether length
C     If noise is added to the data, the program prints to the operator the
C     values of the signal-to-noise ratios on each axis, XSNR and YSNR.
C     The signal-to-noise ratio is calculated as the maximum value divided
C     by the root mean square value for that axis. The noise is calculated
C     using the Box-Muller method of generating Gaussian noise, and a
C     portable random number generator called ran1 found in "Numerical
C     Recipes".

```

```

      REAL X(3000),Y(3000)
C OPEN OUTPUT FILES 'IDFTXY.DAT' (FOR UNOMSC.FOR) AND 'F_REC.DAT' (FOR THE
- C ERROR CALCULATION PROGRAM VTBUSO.FOR)
      OPEN (9,FILE = 'F_REC.DAT',STATUS = 'UNKNOWN')
      OPEN (10,FILE = 'IDFTXY.DAT',STATUS = 'UNKNOWN')
- C READ INPUTS

```

```

PRINT*, 'ENTER THE TIME SAMPLING INTERVAL DT AND BEGINNING TIME'
READ*, DT, TO
PRINT*, 'ENTER NUMBER OF TIME POINTS, STANDARD DEVIATION, RANDOM SEED'
PRINT*, '(RANDOM SEED SHOULD BE A NEGATIVE NUMBER)'
READ*, IPER, SD, ISEED
PRINT*, 'SKIPROPE AMPLITUDE ?, FREQUENCY ?, PHASE ? (FOR X)'
READ*, A1X, F1X, PHI1X
PRINT*, 'SKIPROPE AMPLITUDE ?, FREQUENCY ?, PHASE ? (FOR Y)'
READ*, A1Y, F1Y, PHI1Y
PRINT*, 'NOISE ? YES = 1, NO = 2; % DROPOUT DESIRED (0.0 TO 1.0)'
READ*, NOISE, DROP
PRINT*, 'ENTER ORB RATE, PENDULOUS AMP, FREQ, AND PHASE (FOR X)'
READ*, AOX, A2X, F2X, PHI2X
PRINT*, 'ENTER ORB RATE, PENDULOUS AMP, FREQ, AND PHASE (FOR Y)'
READ*, AOY, A2Y, F2Y, PHI2Y
PRINT*, 'ENTER THE TETHER LENGTH IN KILOMETERS AND AMCSMODE'
READ*, TLNGTH, MODE

```

```

C WRITE THE OUTPUT FILE 'F_REC.DAT'
WRITE(9,*) NOISE, SD, ISEED
WRITE(9,*) A1X, F1X, PHI1X
WRITE(9,*) A1Y, F1Y, PHI1Y
WRITE(9,*) A2X, F2X, PHI2X
C CONVERT PHASES FROM RADIANS TO DEGREES

```

```

PI = 4.0*ATAN(1.0)
CONVRT = PI/180.0
PHI1X = PHI1X*CONVRT
PHI2X = PHI2X*CONVRT
PHI1Y = PHI1Y*CONVRT
PHI2Y = PHI2Y*CONVRT
TPIDT = 2.0*PI*DT

```

```

C ESTIMATE THE SKIPROPE AND PENDULOUS FREQUENCIES FROM THE TETHER LENGTH

```

```

SKIPC1 = 0.2119024
SKIPC2 = -0.3571371
SKIPC3 = 0.5309507
DISCR1 = SKIPC2**2 - 4.0*SKIPC3*(SKIPC1-TLNGTH)
SKIPPR = 40.0*((-1)*SKIPC2 + SQRT(DISCR1))/SKIPC3
PENDC1 = 8.0952965E-02
PENDC2 = 0.5571405
PENDC3 = 0.2476189
DISCR2 = PENDC2**2 - 4.0*PENDC3*(PENDC1-TLNGTH)
PENDFR = 0.01*(1.0+((-1)*PENDC2 + SQRT(DISCR2))/(2.0*PENDC3))
IF (TLNGTH.LT.1.2) THEN
    PENDFR = PENDFR + 0.008
    SKIPPR = 116.049*TLNGTH

```

```

END IF

```

```

SKIPFR = 1.0/SKIPPR

```

```

PRINT*, 'EST. SKIPROPE FREQ = ', SKIPFR, ' EST. PENDULOUS FREQ = ', PENDFR
PRINT*, 'IF YOU WISH TO USE THE EST. FREQ TYPE 1, ELSE TYPE 0'

```

```

READ*, IEST

```

```

IF (IEST.EQ.1) THEN

```

```

-       F1X = SKIPFR
-       F1Y = SKIPFR
-       F2X = PENDFR
-       F2Y = PENDFR
-   END IF

```

```

-   C CALCULATE THE NOISELESS SIGNAL

```

```

-       THE1X = TPIDT*F1X
-       THE2X = TPIDT*F2X
-       THE1Y = TPIDT*F1Y
-       THE2Y = TPIDT*F2Y
-       DO I = 1,IPER
-           I1 = I-1
-           X(I) = A0X + A1X*COS(THE1X*I1+PHI1X)+A2X*COS(THE2X*I1+PHI2X)
-           Y(I) = A0Y + A1Y*COS(THE1Y*I1+PHI1Y)+A2Y*COS(THE2Y*I1+PHI2Y)
-       END DO

```

```

-   C LOOP TO INCORPORATE NOISE INTO THE DATA

```

```

-       IF (NOISE.EQ.2) GO TO 1
-       XRMS = 0.0
-       YRMS = 0.0
-       icount = 0.0
-       XMAX = ABS(X(1))
-       YMAX = ABS(Y(1))
-       DO I = 1,IPER
-           IF (ABS(X(I)).GT.XMAX) XMAX = ABS(X(I))
-           IF (ABS(Y(I)).GT.YMAX) YMAX = ABS(Y(I))
-       2       v1 = 2.0 * ran1(iseed) - 1.0
-           v2 = 2.0 * ran1(iseed) - 1.0
-           r = v1**2 + v2**2
-           if (r.ge.1) go to 2
-           fac = sqrt(-2.0*log(r)/r)
-           xn = v1*fac*sd + x(i)
-           yn = v2*fac*sd + y(i)
-           XRMS = XRMS + (XN - X(I))**2
-           YRMS = YRMS + (YN - Y(I))**2
-           X(I) = XN
-           Y(I) = YN
-       END DO
-       XRMS = SQRT(XRMS/(IPER+1))
-       YRMS = SQRT(YRMS/(IPER+1))
-       XSNR = XMAX/XRMS
-       YSNR = YMAX/YRMS
-       PRINT*, 'XSNR = ', XSNR, ' YSNR = ', YSNR

```

```

-   C LOOP TO SIMULATE DROPOUTS IN THE DATA

```

```

-   1   CONTINUE
-       DROP1 = 1.0 - DROP
-       DO I = 1,IPER
-           IF (ran1(ISEED).GT.DROP1) THEN

```

```

                X(I) = 0.0
                Y(I) = 0.0
            END IF
        END DO
    
```

$$Y(I) = 0.0$$

END IF

END DO

```
C WRITE OUT DATA TO IDFTXY.DAT
```

```
DO I = 1,IPER
  TIME = (I-1)*DT + T0
  WRITE(10,*)TIME,X(I),Y(I),TLNGTH,MODE
END DO
CLOSE (10,STATUS = 'KEEP')
END
```

$$TIME = (I-1)*DT + T0$$

```
WRITE(10,*)TIME,X(I),Y(I),TLNGTH,MODE
```

END DO

```
CLOSE (10,STATUS = 'KEEP')
```

END

```

function ran1(idum)
dimension r(97)
parameter (m1 = 259200, ia1 = 7141, ic1 = 54773, rm1 = 1.0/m1)
parameter (m2 = 134456, ia2 = 8121, ic2 = 28411, rm2 = 1.0/m2)
parameter (m3 = 243000, ia3 = 4561, ic3 = 51349)
data iff /0/
if (idum.lt.0.or.iff.eq.0) then
    iff = 1
    ix1 = mod(ic1 - idum,m1)
    ix1 = mod(ia1*ix1 + ic1,m1)
    ix2 = mod(ix1,m2)
    ix1 = mod(ia1*ix1 + ic1,m1)
    ix3 = mod(ix1,m3)
    do j = 1,97
        ix1 = mod(ia1*ix1 + ic1,m1)
        ix2 = mod(ia2*ix2 + ic2,m2)
        r(j) = (float(ix1) + float(ix2)*rm2)*rm1
    end do
    idum = 1
end if
ix1 = mod(ia1*ix1 + ic1,m1)
ix2 = mod(ia2*ix2 + ic2,m2)
ix3 = mod(ia3*ix3 + ic3,m3)
j = 1 + (97*ix3)/m3
if (j.gt.97.or.j.lt.1) pause
ran1 = r(j)
r(j) = (float(ix1) + float(ix2)*rm2)*rm1
return
end

```

```
dimension r(97)
```

```
dimension r(97)
parameter (m1 = 259200, ia1 = 7141, ic1 = 54773, rm1 = 1.0/m1)
```

```
parameter (m1 = 255200, ia1 = 7212, ic1 = 28411, rm1 = 1.0/m1)
parameter (m2 = 134456, ia2 = 8121, ic2 = 28411, rm2 = 1.0/m2)
```

```
parameter (m2 = 134430, ia2 = 1111, ic2 = 51349)
parameter (m3 = 243000, ia3 = 4561, ic3 = 51349)
```

data iff /0/

```

      data 111,0,
      if (idum.lt.0.or.iff.eq.0) then

```

$$\text{iff} = 1$$

```
ix1 = mod(ic1 - idum,m1)
```

```
ix1 = mod(ia1*ix1 + ic1,m1)
```

```
ix2 = mod(ix1,m2)
```

```
ix1 = mod(ia1*ix1 + ic1,m1)
```

```
ix3 = mod(ix1,m3)
```

do j = 1,97

```
ix1 = mod(ia1*ix1 + ic1,m1)
```

```
ix2 = mod(ia2*ix2 + ic2,m2)
```

```

r(j) = (float(ix1) + float(ix2)*rm2)*rm1

```

end do

idum = 1

```

end if

```

```
ix1 = mod(ia1*ix1 + ic1,m1)
```

```
ix2 = mod(ia2*ix2 + ic2,m2)
```

```
ix3 = mod(ia3*ix3 + ic3,m3)
```

$$j = 1 + (97 * ix3) / m3$$

```
if (j.gt.97.or.j.lt.1) pause
```

```

ran1 = r(j)

```

```

r(j) = (float(ix1) + float(ix2)*rm2)*rm1

```

```

return

```

end


```

- C Program CRELIBR.FOR generates model data sets for the UNOMSC.FOR backup skip-
C rope observer program for the satellite at station 1 with a libration compo-
C nent in the tether. Five numbers are outputted per line of the data file
C IDFTXY.DAT: time, x (roll) axis gyro rate, y (pitch) axis gyro rate, tether
- C length, and amcsmode. The user can opt to include noise and/or dropouts into
C the data. If desired, the program will estimate the skiprope and pendulous
C frequencies as functions of the tether length. (Note: These estimated fre-
- C quencies are exactly that - estimates - and definitely should not be con-
C strued as accurate values!)
C   The operator is prompted to input the following values:
- C       time sampling interval, beginning time tag
C       number of data points, standard deviation, and random seed
C       (standard deviation is 10 X the nominal value of 2.8e-04)
C       (random seed should be a negative seed)
- C       skiprope amplitude, frequency, and phase (for x axis)
C       skiprope amplitude, frequency, and phase (for y axis)
C       1 or 2 to add noise or not, dropout percentage (0.0 to 1.0)
- C       orb rate, pendulous amplitude, frequency, and phase (for x axis)
C       orb rate, pendulous amplitude, frequency, and phase (for y axis)
C       libration amplitude for x, libration amplitude for y, and x phase
C       (y phase is computed as x phase - 90.0)
- C       tether length, amcsmode
C       0 to use inputted values to generate model, 1 to estimate skiprope
C       and pendulous frequencies from the inputted tether length
- C   If noise is added to the data, the program prints to the operator the
C   values of the signal-to-noise ratios on each axis, XSNR and YSNR.
C   The signal-to-noise ratio is calculated as the maximum value divided
- C   by the root mean square value for that axis. The noise is calculated
C   using the Box-Muller method of generating Gaussian noise, and a portable
C   random number generator called ran1 found in "Numerical Recipes".

-   REAL X(3000),Y(3000)
C OPEN OUTPUT FILES 'IDFTXY.DAT' (FOR UNOMSC.FOR) AND 'F_REC.DAT' (FOR THE
C ERROR CALCULATION PROGRAM VTBUSO.FOR)
-   OPEN (9,FILE = 'F_REC.DAT',STATUS = 'UNKNOWN')
C   OPEN (10,FILE = 'IDFTXY.DAT',STATUS = 'UNKNOWN')
C READ INPUTS
-   PRINT*,'ENTER THE TIME SAMPLING INTERVAL DT AND BEGINNING TIME'
C   READ*,DT,TO
-   PRINT*,'ENTER NUMBER OF TIME POINTS, STANDARD DEVIATION, RANDOM SEED'
C   PRINT*,'(RANDOM SEED SHOULD BE A NEGATIVE NUMBER)'
-   READ*,IPER,SD,ISEED
C   PRINT*,'SKIPROPE AMPLITUDE ?, FREQUENCY ?, PHASE ? (FOR X)'
-   READ*,A1X,F1X,PHI1X
C   PRINT*,'SKIPROPE AMPLITUDE ?, FREQUENCY ?, PHASE ? (FOR Y)'
-   READ*,A1Y,F1Y,PHI1Y
C   PRINT*,'NOISE ? YES = 1, NO = 2; % DROPOUT DESIRED (0.0 TO 1.0)'
-   READ*,NOISE,DROP
C   PRINT*,'ENTER ORB RATE, PENDULOUS AMP, FREQ, AND PHASE (FOR X)'
-   READ*,A0X,A2X,F2X,PHI2X
C   PRINT*,'ENTER ORB RATE, PENDULOUS AMP, FREQ, AND PHASE (FOR Y)'
-   READ*,A0Y,A2Y,F2Y,PHI2Y

```

```

PRINT*, 'ENTER LIBRATION AMPLITUDE FOR X, Y, AND PHASE FOR X'
READ*, ALX, ALY, PHILX
PRINT*, 'ENTER THE TETHER LENGTH IN KILOMETERS AND AMCSMODE'
READ*, TLNGTH, MODE

```

```

C WRITE THE OUTPUT FILE 'F_REC.DAT'
WRITE(9,*) NOISE, SD, ISEED
WRITE(9,*) A1X, F1X, PHI1X
WRITE(9,*) A1Y, F1Y, PHI1Y
WRITE(9,*) A2X, F2X, PHI2X

```

```

C CONVERT PHASES FROM RADIANS TO DEGREES AND COMPUTE LIBRATION FREQUENCIES
PHILY = PHILX - 90.0
PI = 4.0*ATAN(1.0)
CONVRT = PI/180.0
PHI1X = PHI1X*CONVRT
PHI2X = PHI2X*CONVRT
PHI1Y = PHI1Y*CONVRT
PHI2Y = PHI2Y*CONVRT
PHILX = PHILX*CONVRT
PHILY = PHILY*CONVRT
TPIDT = 2.0*PI*DT
FLX = 1/2713.0
FLY = 1/3132.0

```

```

C ESTIMATE THE SKIPROPE AND PENDULOUS FREQUENCIES FROM THE TETHER LENGTH
SKIPC1 = 0.2119024
SKIPC2 = -0.3571371
SKIPC3 = 0.5309507
DISCR1 = SKIPC2**2 - 4.0*SKIPC3*(SKIPC1-TLNGTH)
SKIPPR = 40.0*((-1)*SKIPC2 + SQRT(DISCR1))/SKIPC3
PENDC1 = 8.0952965E-02
PENDC2 = 0.5571405
PENDC3 = 0.2476189
DISCR2 = PENDC2**2 - 4.0*PENDC3*(PENDC1-TLNGTH)
PENDFR = 0.01*(1.0+((-1)*PENDC2 + SQRT(DISCR2))/(2.0*PENDC3))
IF (TLNGTH.LT.1.2) THEN
    PENDFR = PENDFR + 0.008
    SKIPPR = 116.049*TLNGTH
END IF
SKIPFR = 1.0/SKIPPR
PRINT*, 'EST. SKIPROPE FREQ = ', SKIPFR, ' EST. PENDULOUS FREQ = ', PENDFR
PRINT*, 'IF YOU WISH TO USE THE EST. FREQ TYPE 1, ELSE TYPE 0'
READ*, IEST
IF (IEST.EQ.1) THEN
    F1X = SKIPFR
    F1Y = SKIPFR
    F2X = PENDFR
    F2Y = PENDFR
END IF

```

```

- C CALCULATE THE NOISELESS SIGNAL
  THE1X = TPIDT*F1X
  THE2X = TPIDT*F2X
  THE1Y = TPIDT*F1Y
  THE2Y = TPIDT*F2Y
  THE1X = TPIDT*FLX
  THE1Y = TPIDT*FLY
  DO I = 1,IPER
    I1 = I-1
    X(I) = A0X + A1X*COS(THE1X*I1+PHI1X)+A2X*COS(THE2X*I1+PHI2X)
    #      + ALX*COS(THELX*I1+PHILX)
    Y(I) = A0Y + A1Y*COS(THE1Y*I1+PHI1Y)+A2Y*COS(THE2Y*I1+PHI2Y)
    &      + ALY*COS(THELY*I1+PHILY)
  END DO

```

```

- C LOOP TO INCORPORATE NOISE INTO THE DATA

```

```

- IF (NOISE.EQ.2) GO TO 1
  XRMS = 0.0
  YRMS = 0.0
  icount = 0.0
  XMAX = ABS(X(1))
  YMAX = ABS(Y(1))
  DO I = 1,IPER
    IF (ABS(X(I)).GT.XMAX) XMAX = ABS(X(I))
    IF (ABS(Y(I)).GT.YMAX) YMAX = ABS(Y(I))
    2  v1 = 2.0 * ran1(iseed) - 1.0
    v2 = 2.0 * ran1(iseed) - 1.0
    r = v1**2 + v2**2
    if (r.ge.1) go to 2
    fac = sqrt(-2.0*log(r)/r)
    xn = v1*fac*sd + x(i)
    yn = v2*fac*sd + y(i)
    XRMS = XRMS + (XN - X(I))**2
    YRMS = YRMS + (YN - Y(I))**2
    X(I) = XN
    Y(I) = YN
  END DO
  XRMS = SQRT(XRMS/(IPER+1))
  YRMS = SQRT(YRMS/(IPER+1))
  XSNR = XMAX/XRMS
  YSNR = YMAX/YRMS
  PRINT*, 'XSNR = ', XSNR, ' YSNR = ', YSNR

```

```

- C LOOP TO SIMULATE DROPOUTS IN THE DATA

```

```

  1  CONTINUE
  DROP1 = 1.0 - DROP
  DO I = 1,IPER
    IF (ran1(ISEED).GT.DROP1) THEN
      X(I) = 0.0
      Y(I) = 0.0
    
```

```
END IF
END DO
```

```
C WRITE OUT DATA TO IDFTXY.DAT
```

```
DO I = 1,IPER
  TIME = (I-1)*DT + T0
  WRITE(10,*)TIME,X(I),Y(I),TLNGTH,MODE
END DO
CLOSE (10,STATUS = 'KEEP')
END
```

```
function ran1(idum)
dimension r(97)
parameter (m1 = 259200, ia1 = 7141, ic1 = 54773, rm1 = 1.0/m1)
parameter (m2 = 134456, ia2 = 8121, ic2 = 28411, rm2 = 1.0/m2)
parameter (m3 = 243000, ia3 = 4561, ic3 = 51349)
data iff /0/
if (idum.lt.0.or.iff.eq.0) then
  iff = 1
  ix1 = mod(ic1 - idum,m1)
  ix1 = mod(ia1*ix1 + ic1,m1)
  ix2 = mod(ix1,m2)
  ix1 = mod(ia1*ix1 + ic1,m1)
  ix3 = mod(ix1,m3)
  do j = 1,97
    ix1 = mod(ia1*ix1 + ic1,m1)
    ix2 = mod(ia2*ix2 + ic2,m2)
    r(j) = (float(ix1) + float(ix2)*rm2)*rm1
  end do
  idum = 1
end if
ix1 = mod(ia1*ix1 + ic1,m1)
ix2 = mod(ia2*ix2 + ic2,m2)
ix3 = mod(ia3*ix3 + ic3,m3)
j = 1 + (97*ix3)/m3
if (j.gt.97.or.j.lt.1) pause
ran1 = r(j)
r(j) = (float(ix1) + float(ix2)*rm2)*rm1
return
end
```

APPENDIX 2.B

Results of ECR Verification Table 2. Filter Test Cases.

This is the F_YAWMAN.DAT file for case 1 of the verification table.

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04

REV LABEL	POLARITY	RATE(D/S)	# OF ROTATIONS	START TIME
0	POSITIVE	1.9439	.8	1143.32
1	POSITIVE	1.9439	.8	1328.52
2	POSITIVE	1.9439	.8	1513.72
3	POSITIVE	1.9439	.8	1698.91
4	POSITIVE	1.9439	.8	1884.11

This is the T_REPORT.DAT file for case 1 of the verification table.

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.02001	-.041 %		
Amp y	.02000	.01999	.063 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000000 Hz		
Phase x	-107.0	-106.7	.32	.054	.58
Phase y	163.0	163.3	.35	.010	.39
Pendulous data, amp, freq, phase			=:	.000	.03125 10.0
Equivalent U & V deflections (meters)			=:	7.9	7.9
No noise for this run.					
Stop index, start index, & number of point = 593 1 593					

The following eleven pages represent the results of case 2 of the verification table. The first ten pages give the results of the ten individual noise runs, with the F_YAWMAN.DAT file listed first and then the T_REPORT.DAT file. The last page lists the averaged results of the T_REPORT.DAT files.

- MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9436 .8 1144.53
 1 POSITIVE 1.9436 .8 1329.76
 2 POSITIVE 1.9436 .8 1514.98
 3 POSITIVE 1.9436 .8 1700.21
 4 POSITIVE 1.9436 .8 1885.44

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01989	.565 %		
Amp y	.02000	.01955	2.226 %		
Freq x	.00540	.00540	.000005 Hz		
Freq y	.00540	.00539	.000007 Hz		
Phase x	-107.0	-107.5	.51	.325	2.09
Phase y	163.0	165.2	2.16	.489	4.53
Pendulous data, amp, freq, phase			=:	.000	.03125 10.0
Equivalent U & V deflections (meters)			=:	7.7	7.8
iseed =	-1000				
Noise data:	1 sigma value (deg/sec)			.000280	
Stop index, start index, & number of point			= 593	1 593	

- MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 - 0 POSITIVE 1.9452 .8 1143.61
 1 POSITIVE 1.9452 .8 1328.68
 2 POSITIVE 1.9452 .8 1513.75
 3 POSITIVE 1.9452 .8 1698.82
 - 4 POSITIVE 1.9452 .8 1883.89

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.02005	-.230 %		
Amp y	.02000	.01975	1.239 %		
Freq x	.00540	.00540	.000003 Hz		
Freq y	.00540	.00540	.000004 Hz		
Phase x	-107.0	-107.4	.38	.169	1.20
Phase y	163.0	163.3	.34	.279	1.70
Pendulous data, amp, freq, phase			=:	.000	.03125 10.0
Equivalent U & V deflections (meters)			=:	7.8	7.9
iseed =	-2000				
Noise data:	1 sigma value (deg/sec)			.000280	
Stop index, start index, & number of point			= 593	1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9444 .8 1143.05
 1 POSITIVE 1.9444 .8 1328.20
 2 POSITIVE 1.9444 .8 1513.35
 3 POSITIVE 1.9444 .8 1698.50
 4 POSITIVE 1.9444 .8 1883.65

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01974	1.322 %		
Amp y	.02000	.02024	-1.204 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	-107.0	-106.9	.10	.084	.50
Phase y	163.0	163.1	.08	.056	.35
Pendulous data, amp, freq, phase	=:			.000	.03125 10.0
Equivalent U & V deflections (meters)	=:			8.0	7.8
iseed =	-3000				
Noise data: 1 sigma value (deg/sec)	:			.000280	
Stop index, start index, & number of point	=	593	1	593	

- MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9440 .8 1144.27
 1 POSITIVE 1.9440 .8 1329.46
 2 POSITIVE 1.9440 .8 1514.64
 3 POSITIVE 1.9440 .8 1699.82
 4 POSITIVE 1.9440 .8 1885.01

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.02024	-1.224 %		
Amp y	.02000	.01972	1.384 %		
Freq x	.00540	.00540	.000002 Hz		
Freq y	.00540	.00540	.000002 Hz		
Phase x	-107.0	-107.3	.33	.139	1.01
Phase y	163.0	164.2	1.20	.132	1.84
pendulous data, amp, freq, phase =:				.000	.03125 10.0
Equivalent U & V deflections (meters) =:				7.8	8.0
iseed = -4000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

- MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9405 .8 1145.21
 1 POSITIVE 1.9405 .8 1330.73
 2 POSITIVE 1.9405 .8 1516.24
 3 POSITIVE 1.9405 .8 1701.76
 4 POSITIVE 1.9405 .8 1887.28

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01991	.448 %		
Amp y	.02000	.02040	-2.024 %		
Freq x	.00540	.00539	.000009 Hz		
Freq y	.00540	.00539	.000010 Hz		
Phase x	-107.0	-107.5	.52	.610	3.48
Phase y	163.0	164.8	1.85	.679	5.15
Pendulous data, amp, freq, phase	=:	.000	.03125	10.0	
Equivalent U & V deflections (meters)	=:	8.0	7.8		
iseed =	-5000				
Noise data: 1 sigma value (deg/sec) :		.000280			
Stop index, start index, & number of point =	593	1	593		

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9412 .8 1144.85
 1 POSITIVE 1.9412 .8 1330.31
 2 POSITIVE 1.9412 .8 1515.76
 3 POSITIVE 1.9412 .8 1701.22
 4 POSITIVE 1.9412 .8 1886.68

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.02028	-1.423 %		
Amp y	.02000	.02004	-.193 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00538	.000016 Hz		
Phase x	-107.0	-106.3	.73	.044	.95
Phase y	163.0	164.8	1.80	1.097	7.13
Pendulous data, amp, freq, phase			=:	.000	.03125 10.0
Equivalent U & V deflections (meters)			=:	7.9	8.0
iseed = -6000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

- MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 - 0 POSITIVE 1.9405 .8 1144.19
 1 POSITIVE 1.9405 .8 1329.71
 2 POSITIVE 1.9405 .8 1515.23
 3 POSITIVE 1.9405 .8 1700.75
 - 4 POSITIVE 1.9405 .8 1886.27

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01992	.383 %		
Amp y	.02000	.01996	.184 %		
Freq x	.00540	.00539	.000008 Hz		
Freq y	.00540	.00539	.000012 Hz		
Phase x	-107.0	-105.6	1.42	.502	3.86
Phase y	163.0	164.8	1.79	.791	5.64
Pendulous data, amp, freq, phase =:				.000	.03125 10.0
Equivalent U & V deflections (meters) =:				7.9	7.8
iseed = -7000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =				593 1 593	

- MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9454 .8 1142.47
 1 POSITIVE 1.9454 .8 1327.52
 2 POSITIVE 1.9454 .8 1512.57
 3 POSITIVE 1.9454 .8 1697.62
 4 POSITIVE 1.9454 .8 1882.67

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01985	.737 %		
Amp y	.02000	.02034	-1.681 %		
Freq x	.00540	.00539	.000005 Hz		
Freq y	.00540	.00541	.000013 Hz		
Phase x	-107.0	-107.4	.39	.357	2.12
Phase y	163.0	162.0	1.01	.880	5.28
Pendulous data, amp, freq, phase	=:			.000	.03125 10.0
Equivalent U & V deflections (meters)	=:			8.0	7.8
iseed =	-8000				
Noise data: 1 sigma value (deg/sec)				.000280	
Stop index, start index, & number of point			593	1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03

TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04

REV LABEL	POLARITY	RATE(D/S)	# OF ROTATIONS	START TIME
0	POSITIVE	1.9464	.8	1142.96
1	POSITIVE	1.9464	.8	1327.91
2	POSITIVE	1.9464	.8	1512.87
3	POSITIVE	1.9464	.8	1697.83
4	POSITIVE	1.9464	.8	1882.79

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01977	1.143 %		
Amp y	.02000	.02004	-.209 %		
Freq x	.00540	.00541	.000008 Hz		
Freq y	.00540	.00541	.000006 Hz		
Phase x	-107.0	-108.3	1.29	.501	3.73
Phase y	163.0	163.2	.20	.379	2.04
Pendulous data, amp, freq, phase	=:		.000	.03125	10.0
Equivalent U & V deflections (meters)	=:		7.9	7.8	
iseed =	-9000				
Noise data:	1 sigma value (deg/sec) :		.000280		
Stop index, start index, & number of point	= 593		1 593		

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9456 .8 1142.38
 1 POSITIVE 1.9456 .8 1327.41
 2 POSITIVE 1.9456 .8 1512.45
 3 POSITIVE 1.9456 .8 1697.48
 4 POSITIVE 1.9456 .8 1882.52

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01999	.069 %		
Amp y	.02000	.02025	-1.240 %		
Freq x	.00540	.00540	.000004 Hz		
Freq y	.00540	.00540	.000005 Hz		
Phase x	-107.0	-106.5	.45	.256	1.70
Phase y	163.0	163.3	.28	.328	1.87
Pendulous data, amp, freq, phase			=:	.000	.03125 10.0
Equivalent U & V deflections (meters)			=:	8.0	7.8
iseed =	-10000				
Noise data:		1 sigma value (deg/sec)	:	.000280	
Stop index, start index, & number of point		= 593		1 593	

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01996	.179 %		
Amp y	.02000	.02003	-.152 %		
Freq x	.00540	.00540	.000005 Hz		
Freq y	.00540	.00540	.000008 Hz		
Phase x	-107.0	-107.1	.61	.299	2.06
Phase y	163.0	163.9	1.07	.511	3.55
Pendulous data, amp, freq, phase			=:	.000	.03125 10.0
Equivalent U & V deflections (meters)			=:	7.9	7.8
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =			593	1 593	

This is the F_YWAMAN.DAT file for case 3 of the verification table.

REV LABEL	POLARITY	RATE(D/S)	# OF ROTATIONS	START TIME
0	POSITIVE	1.6581	.9	1028.70
1	POSITIVE	1.6581	.9	1245.81
2	POSITIVE	1.6581	.9	1462.92
3	POSITIVE	1.6581	.9	1680.03
4	POSITIVE	1.6581	.9	1897.14

This is the T_REPORT.DAT file for case 3 of the verification table.

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.02002	-.099 %		
Amp y	.02000	.01995	.269 %		
Freq x	.00460	.00460	.000003 Hz		
Freq y	.00460	.00461	.000009 Hz		
Phase x	50.0	49.2	.80	.226	1.74
Phase y	-40.0	-41.3	1.28	.705	4.20
Pendulous data, amp, freq, phase			=:	.500	.03125 60.0
Equivalent U & V deflections (meters)			=:	9.2	9.2
No noise for this run.					
Stop index, start index, & number of point				593	1 593

The following eleven pages represent the results of case 4 of the verification table. The first ten pages give the results of the ten individual noise runs, with the F_YAWMAN.DAT file listed first and then the T_REPORT.DAT file. The last page lists the averaged results of the T_REPORT.DAT files.

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.6565 .9 1029.51
 1 POSITIVE 1.6565 .9 1246.83
 2 POSITIVE 1.6565 .9 1464.15
 3 POSITIVE 1.6565 .9 1681.47
 4 POSITIVE 1.6565 .9 1898.79

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01989	.556 %		
Amp y	.02000	.02014	-.676 %		
Freq x	.00460	.00460	.000001 Hz		
Freq y	.00460	.00460	.000004 Hz		
Phase x	50.0	49.4	.61	.044	.79
Phase y	-40.0	-40.7	.72	.281	1.88
Pendulous data, amp, freq, phase =:			.500	.03125	60.0
Equivalent U & V deflections (meters) =:			9.3	9.2	
iseed = -100000					
Noise data: 1 sigma value (deg/sec) :			.000280		
Stop index, start index, & number of point = 593			1 593		

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.6618 .9 1026.85
 1 POSITIVE 1.6618 .9 1243.48
 2 POSITIVE 1.6618 .9 1460.11
 3 POSITIVE 1.6618 .9 1676.75
 4 POSITIVE 1.6618 .9 1893.38

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.02009	-.443 %		
Amp y	.02000	.02020	-1.014 %		
Freq x	.00460	.00462	.000016 Hz		
Freq y	.00460	.00462	.000016 Hz		
Phase x	50.0	48.1	1.94	1.244	7.09
Phase y	-40.0	-42.5	2.49	1.276	7.77
Pendulous data, amp, freq, phase			=:	.500	.03125 60.0
Equivalent U & V deflections (meters) =:				9.3	9.2
iseed = -200000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =				593	1 593

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.6579 .9 1029.86
 1 POSITIVE 1.6579 .9 1247.00
 2 POSITIVE 1.6579 .9 1464.15
 3 POSITIVE 1.6579 .9 1681.30
 4 POSITIVE 1.6579 .9 1898.44

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01980	.979 %		
Amp y	.02000	.01971	1.441 %		
Freq x	.00460	.00460	.000000 Hz		
Freq y	.00460	.00461	.000010 Hz		
Phase x	50.0	48.8	1.18	.001	1.19
Phase y	-40.0	-40.5	.51	.814	3.88
Pendulous data, amp, freq, phase			=:	.500	.03125 60.0
Equivalent U & V deflections (meters)			=:	9.1	9.1
iseed = -300000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.6587 .9 1029.43
 1 POSITIVE 1.6587 .9 1246.47
 2 POSITIVE 1.6587 .9 1463.51
 3 POSITIVE 1.6587 .9 1680.54
 4 POSITIVE 1.6587 .9 1897.58

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01973	1.347 %		
Amp y	.02000	.02016	-.799 %		
Freq x	.00460	.00461	.000010 Hz		
Freq y	.00460	.00460	.000005 Hz		
Phase x	50.0	47.7	2.29	.819	5.68
Phase y	-40.0	-40.8	.80	.357	2.28
Pendulous data, amp, freq, phase			=:	.500	.03125 60.0
Equivalent U & V deflections (meters) =:				9.3	9.1
iseed = -400000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.6575 .9 1029.04
 1 POSITIVE 1.6575 .9 1246.24
 2 POSITIVE 1.6575 .9 1463.44
 3 POSITIVE 1.6575 .9 1680.63
 4 POSITIVE 1.6575 .9 1897.83

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.02001	-.071 %		
Amp y	.02000	.02022	-1.082 %		
Freq x	.00460	.00461	.000007 Hz		
Freq y	.00460	.00460	.000001 Hz		
Phase x	50.0	49.0	1.04	.570	3.40
Phase y	-40.0	-40.8	.83	.068	1.11
Pendulous data, amp, freq, phase			=:	.500	.03125 60.0
Equivalent U & V deflections (meters)			=:	9.3	9.2
iseed =	-500000				
Noise data:	1 sigma value (deg/sec)			.000280	
Stop index, start index, & number of point	593	1 593			

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.6582 .9 1028.65
 1 POSITIVE 1.6582 .9 1245.75
 2 POSITIVE 1.6582 .9 1462.85
 3 POSITIVE 1.6582 .9 1679.95
 4 POSITIVE 1.6582 .9 1897.05

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.02017	-.845 %		
Amp y	.02000	.01987	.649 %		
Freq x	.00460	.00461	.000013 Hz		
Freq y	.00460	.00460	.000000 Hz		
Phase x	50.0	49.5	.49	.996	4.62
Phase y	-40.0	-40.1	.08	.027	.20
Pendulous data, amp, freq, phase			=:	.500	.03125 60.0
Equivalent U & V deflections (meters)			=:	9.2	9.3
iseed = -600000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04

REV LABEL	POLARITY	RATE(D/S)	# OF ROTATIONS	START TIME
0	POSITIVE	1.6577	.9	1029.96
1	POSITIVE	1.6577	.9	1247.13
2	POSITIVE	1.6577	.9	1464.30
3	POSITIVE	1.6577	.9	1681.47
4	POSITIVE	1.6577	.9	1898.65

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01965	1.757 %		
Amp y	.02000	.02015	-.735 %		
Freq x	.00460	.00461	.000012 Hz		
Freq y	.00460	.00460	.000003 Hz		
Phase x	50.0	47.9	2.06	.930	5.91
Phase y	-40.0	-40.3	.26	.203	1.11
Pendulous data, amp, freq, phase =:				.500	.03125 60.0
Equivalent U & V deflections (meters) =:				9.3	9.1
iseed = -700000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.6561 .9 1029.72
 1 POSITIVE 1.6561 .9 1247.10
 2 POSITIVE 1.6561 .9 1464.47
 3 POSITIVE 1.6561 .9 1681.85
 4 POSITIVE 1.6561 .9 1899.22

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.01974	1.309 %		
Amp y	.02000	.01975	1.247 %		
Freq x	.00460	.00460	.000000 Hz		
Freq y	.00460	.00460	.000001 Hz		
Phase x	50.0	49.6	.39	.030	.51
Phase y	-40.0	-41.4	1.42	.083	1.76
Pendulous data, amp, freq, phase			=:	.500	.03125 60.0
Equivalent U & V deflections (meters) =:				9.1	9.1
iseed = -800000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.6571 .9 1029.24
 1 POSITIVE 1.6571 .9 1246.49
 2 POSITIVE 1.6571 .9 1463.74
 3 POSITIVE 1.6571 .9 1681.00
 4 POSITIVE 1.6571 .9 1898.25

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.02022	-1.115 %		
Amp y	.02000	.01993	.348 %		
Freq x	.00460	.00460	.000001 Hz		
Freq y	.00460	.00461	.000007 Hz		
Phase x	50.0	49.8	.22	.111	.68
Phase y	-40.0	-40.8	.84	.575	3.22
Pendulous data, amp, freq, phase			=:	.500	.03125 60.0
Equivalent U & V deflections (meters) =:				9.2	9.3
iseed =	-900000				
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =			593	1	593

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.6632 .9 1026.13
 1 POSITIVE 1.6632 .9 1242.58
 2 POSITIVE 1.6632 .9 1459.03
 3 POSITIVE 1.6632 .9 1675.48
 4 POSITIVE 1.6632 .9 1891.93

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.02000	.02002	-.113 %		
Amp y	.02000	.02016	-.806 %		
Freq x	.00460	.00463	.000026 Hz		
Freq y	.00460	.00461	.000014 Hz		
Phase x	50.0	47.5	2.53	2.070	11.10
Phase y	-40.0	-40.7	.66	1.062	5.06
Pendulous data, amp, freq, phase =:			.500	.03125	60.0
Equivalent U & V deflections (meters) =:			9.3	9.2	
iseed = -1000000					
Noise data: 1 sigma value (deg/sec) :			.000280		
Stop index, start index, & number of point = 593			1 593		

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
--	-------	--------	-------	--------------------------	--------------------------

Amp x	.02000	.01993	.336 %		
Amp y	.02000	.02003	-.143 %		

Freq x	.00460	.00461	.000009 Hz		
Freq y	.00460	.00461	.000006 Hz		

Phase x	50.0	48.7	1.28	.682	4.10
Phase y	-40.0	-40.9	.86	.475	2.83

Pendulous data, amp, freq, phase	=:	.500	.03125	60.0
Equivalent U & V deflections (meters) =:	9.2	9.2		
Noise data: 1 sigma value (deg/sec) :		.000280		
Stop index, start index, & number of point =	593	1	593	

The following eleven pages represent the results of case 5 of the verification table. The first ten pages give the results of the ten individual noise runs, with the F_YAWMAN.DAT file listed first and then the T_REPORT.DAT file. The last page lists the averaged results of the T_REPORT.DAT files.

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9436 5.9 1190.58
 1 POSITIVE 1.9436 5.9 1375.80
 2 POSITIVE 1.9436 5.9 1561.02
 3 POSITIVE 1.9436 5.9 1746.24
 4 POSITIVE 1.9436 5.9 1931.46

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MI degs
Amp x	.15000	.14993	.048 %		
Amp y	.15000	.15022	-.146 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.5	.53	.073	.89
Phase y	70.0	70.4	.37	.068	.70
Pendulous data, amp, freq, phase			=:	.000	.03125 50.0
Equivalent U & V deflections (meters)			=:	59.0	58.9
iseed = -11000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =			593	1	593

- MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 - 0 POSITIVE 1.9437 5.9 1190.56
 1 POSITIVE 1.9437 5.9 1375.78
 2 POSITIVE 1.9437 5.9 1560.99
 3 POSITIVE 1.9437 5.9 1746.21
 - 4 POSITIVE 1.9437 5.9 1931.43

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.15003	-.020 %		
Amp y	.15000	.15015	-.100 %		
Freq x	.00540	.00540	.000000 Hz		
Freq y	.00540	.00540	.000002 Hz		
Phase x	160.0	160.4	.42	.026	.54
Phase y	70.0	70.5	.53	.102	1.03
pendulous data, amp, freq, phase			=:	.000	.03125 50.0
Equivalent U & V deflections (meters)			=:	59.0	59.0
iseed = -12000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =			593	1	593

- MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 - 0 POSITIVE 1.9434 5.9 1190.72
 - 1 POSITIVE 1.9434 5.9 1375.97
 2 POSITIVE 1.9434 5.9 1561.22
 3 POSITIVE 1.9434 5.9 1746.46
 - 4 POSITIVE 1.9434 5.9 1931.71

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.15011	-.074 %		
Amp y	.15000	.15012	-.082 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000002 Hz		
Phase x	160.0	160.4	.43	.074	.79
Phase y	70.0	70.6	.59	.164	1.39
Pendulous data, amp, freq, phase			=:	.000	.03125 50.0
Equivalent U & V deflections (meters)			=:	59.0	59.0
iseed = -13000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =			593	1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9436 5.9 1190.57
 1 POSITIVE 1.9436 5.9 1375.79
 2 POSITIVE 1.9436 5.9 1561.02
 3 POSITIVE 1.9436 5.9 1746.24
 4 POSITIVE 1.9436 5.9 1931.46

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14979	.140 %		
Amp y	.15000	.14998	.013 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.5	.47	.093	.92
Phase y	70.0	70.2	.23	.046	.46
Pendulous data, amp, freq, phase			=:	.000	.03125 50.0
Equivalent U & V deflections (meters)			=:	59.0	58.9
iseed =	-14000				
Noise data:	1 sigma value (deg/sec)			.000280	
Stop index, start index, & number of point			= 593	1 593	

- MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 - 0 POSITIVE 1.9438 5.9 1190.49
 - 1 POSITIVE 1.9438 5.9 1375.69
 - 2 POSITIVE 1.9438 5.9 1560.90
 - 3 POSITIVE 1.9438 5.9 1746.10
 - 4 POSITIVE 1.9438 5.9 1931.31

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.15010	-.068 %		
Amp y	.15000	.15037	-.246 %		
Freq x	.00540	.00540	.000000 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.2	.24	.013	.31
Phase y	70.0	70.6	.59	.094	1.05
Pendulous data, amp, freq, phase =:			.000	.03125	50.0
Equivalent U & V deflections (meters) =:			59.1	59.0	
iseed = -15000					
Noise data: 1 sigma value (deg/sec) :			.000280		
Stop index, start index, & number of point = 593			1	593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03

TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04

REV LABEL	POLARITY	RATE(D/S)	# OF ROTATIONS	START TIME
0	POSITIVE	1.9446	5.9	1190.04
1	POSITIVE	1.9446	5.9	1375.18
2	POSITIVE	1.9446	5.9	1560.31
3	POSITIVE	1.9446	5.9	1745.44
4	POSITIVE	1.9446	5.9	1930.57

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
--	-------	--------	-------	--------------------------	--------------------------

Amp x	.15000	.14995	.031 %		
Amp y	.15000	.15009	-.060 %		

Freq x	.00540	.00540	.000003 Hz		
Freq y	.00540	.00540	.000001 Hz		

Phase x	160.0	160.0	.01	.168	.82
Phase y	70.0	70.1	.13	.043	.34

Pendulous data, amp, freq, phase =: .000 .03125 50.0
Equivalent U & V deflections (meters) =: 59.0 58.9
iseed = -16000

Noise data: 1 sigma value (deg/sec) : .000280
Stop index, start index, & number of point = 593 1 593

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9443 5.9 1190.21
 1 POSITIVE 1.9443 5.9 1375.37
 2 POSITIVE 1.9443 5.9 1560.53
 3 POSITIVE 1.9443 5.9 1745.69
 4 POSITIVE 1.9443 5.9 1930.84

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14987	.084 %		
Amp y	.15000	.14994	.040 %		
Freq x	.00540	.00540	.000000 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.5	.50	.012	.56
Phase y	70.0	70.1	.14	.090	.58

Pendulous data, amp, freq, phase =: .000 .03125 50.0
 Equivalent U & V deflections (meters) =: 58.9 58.9
 iseed = -17000
 Noise data: 1 sigma value (deg/sec) : .000280
 Stop index, start index, & number of point = 593 1 593

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9437 5.9 1190.51
 1 POSITIVE 1.9437 5.9 1375.72
 2 POSITIVE 1.9437 5.9 1560.93
 3 POSITIVE 1.9437 5.9 1746.14
 4 POSITIVE 1.9437 5.9 1931.35

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14977	.152 %		
Amp y	.15000	.14986	.096 %		
Freq x	.00540	.00540	.000000 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.5	.55	.029	.69
Phase y	70.0	70.2	.19	.067	.51
Pendulous data, amp, freq, phase			=:	.000	.03125 50.0
Equivalent U & V deflections (meters)			=:	58.9	58.9
iseed =	-18000				
Noise data:	1 sigma value (deg/sec)			.000280	
Stop index, start index, & number of point			= 593	1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9440 5.9 1190.34
 1 POSITIVE 1.9440 5.9 1375.52
 2 POSITIVE 1.9440 5.9 1560.70
 3 POSITIVE 1.9440 5.9 1745.88
 4 POSITIVE 1.9440 5.9 1931.06

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14949	.343 %		
Amp y	.15000	.14994	.038 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.2	.22	.067	.55
Phase y	70.0	70.3	.27	.049	.51
Pendulous data, amp, freq, phase	=:	.000	.03125	50.0	
Equivalent U & V deflections (meters)	=:	58.9	58.7		
iseed =	-19000				
Noise data: 1 sigma value (deg/sec) :		.000280			
Stop index, start index, & number of point =	593	1 593			

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9439 5.9 1190.42
 1 POSITIVE 1.9439 5.9 1375.61
 2 POSITIVE 1.9439 5.9 1560.80
 3 POSITIVE 1.9439 5.9 1746.00
 4 POSITIVE 1.9439 5.9 1931.19

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14975	.168 %		
Amp y	.15000	.15013	-.089 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.2	.19	.045	.41
Phase y	70.0	70.2	.25	.080	.64
Pendulous data, amp, freq, phase			=:	.000	.03125 50.0
Equivalent U & V deflections (meters)			=:	59.0	58.8
iseed = -20000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14988	.080 %		
Amp y	.15000	.15008	-.054 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.3	.36	.060	.65
Phase y	70.0	70.3	.33	.080	.72
Pendulous data, amp, freq, phase =:				.000	.03125 50.0
Equivalent U & V deflections (meters) =:				59.0	58.9
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =				593 1 593	

The following eleven pages represent the results of case 6 of the verification table. The first ten pages give the results of the ten individual noise runs, with the F_YAWMAN.DAT file listed first and then the T_REPORT.DAT file. The last page lists the averaged results of the T_REPORT.DAT files.

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9441 5.9 1190.31
 1 POSITIVE 1.9441 5.9 1375.49
 2 POSITIVE 1.9441 5.9 1560.67
 3 POSITIVE 1.9441 5.9 1745.84
 4 POSITIVE 1.9441 5.9 1931.02

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14984	.104 %		
Amp y	.15000	.15014	-.096 %		
Freq x	.00540	.00540	.000000 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.4	.45	.014	.51
Phase y	70.0	70.1	.05	.048	.28
Pendulous data, amp, freq, phase			=:	.500	.03125 50.0
Equivalent U & V deflections (meters)			=:	59.0	58.9
iseed = -110000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =			593	1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9436 5.9 1190.61
 1 POSITIVE 1.9436 5.9 1375.84
 2 POSITIVE 1.9436 5.9 1561.07
 3 POSITIVE 1.9436 5.9 1746.29
 4 POSITIVE 1.9436 5.9 1931.52

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MI degs
Amp x	.15000	.14932	.451 %		
Amp y	.15000	.14999	.007 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.4	.45	.069	.79
Phase y	70.0	70.3	.31	.096	.78
Pendulous data, amp, freq, phase			=:	.500	.03125 50.0
Equivalent U & V deflections (meters)			=:	59.0	58.7
iseed = -120000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =			593	1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04

REV LABEL	POLARITY	RATE(D/S)	# OF ROTATIONS	START TIME
0	POSITIVE	1.9437	5.9	1190.51
1	POSITIVE	1.9437	5.9	1375.72
2	POSITIVE	1.9437	5.9	1560.93
3	POSITIVE	1.9437	5.9	1746.14
4	POSITIVE	1.9437	5.9	1931.35

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14946	.357 %		
Amp y	.15000	.14998	.016 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000002 Hz		
Phase x	160.0	160.3	.27	.055	.53
Phase y	70.0	70.5	.46	.150	1.19
Pendulous data, amp, freq, phase =:			.500	.03125	50.0
Equivalent U & V deflections (meters) =:			58.9	58.7	
iseed = -130000					
Noise data: 1 sigma value (deg/sec) :			.000280		
Stop index, start index, & number of point =			593	1	593

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9436 5.9 1190.60
 1 POSITIVE 1.9436 5.9 1375.82
 2 POSITIVE 1.9436 5.9 1561.05
 3 POSITIVE 1.9436 5.9 1746.27
 4 POSITIVE 1.9436 5.9 1931.50

	INPUT	OUTPUT	ERROR	PHASE GRAD deg/cycle	PHASE @ T=15 MIN deg
Amp x	.15000	.14980	.136 %		
Amp y	.15000	.15020	-.134 %		
Freq x	.00540	.00540	.000003 Hz		
Freq y	.00540	.00540	.000000 Hz		
Phase x	160.0	160.5	.46	.181	1.33
Phase y	70.0	70.1	.15	.025	.27
Pendulous data, amp, freq, phase			=:	.500	.03125 50.0
Equivalent U & V deflections (meters)			=:	59.0	58.9
iseed = -140000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =			593	1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04

REV LABEL	POLARITY	RATE(D/S)	# OF ROTATIONS	START TIME
0	POSITIVE	1.9439	5.9	1190.45
1	POSITIVE	1.9439	5.9	1375.65
2	POSITIVE	1.9439	5.9	1560.85
3	POSITIVE	1.9439	5.9	1746.04
4	POSITIVE	1.9439	5.9	1931.24

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MI degs
Amp x	.15000	.14995	.033 %		
Amp y	.15000	.15008	-.053 %		
Freq x	.00540	.00540	.000000 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.4	.37	.019	.46
Phase y	70.0	70.3	.28	.075	.64
Pendulous data, amp, freq, phase			=:	.500	.03125 50.0
Equivalent U & V deflections (meters)			=:	59.0	58.9
iseed = -150000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =			593	1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04

REV LABEL	POLARITY	RATE(D/S)	# OF ROTATIONS	START TIME
0	POSITIVE	1.9439	5.9	1190.44
1	POSITIVE	1.9439	5.9	1375.64
2	POSITIVE	1.9439	5.9	1560.83
3	POSITIVE	1.9439	5.9	1746.03
4	POSITIVE	1.9439	5.9	1931.23

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14975	.165 %		
Amp y	.15000	.14999	.009 %		
Freq x	.00540	.00540	.000000 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.3	.27	.022	.37
Phase y	70.0	70.4	.40	.071	.74
Pendulous data, amp, freq, phase	=:	.500	.03125	50.0	
Equivalent U & V deflections (meters)	=:	58.9	58.9		
iseed =	-160000				
Noise data: 1 sigma value (deg/sec)	:	.000280			
Stop index, start index, & number of point	=	593	1	593	

- MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 - 0 POSITIVE 1.9439 5.9 1190.44
 - 1 POSITIVE 1.9439 5.9 1375.64
 2 POSITIVE 1.9439 5.9 1560.84
 3 POSITIVE 1.9439 5.9 1746.04
 - 4 POSITIVE 1.9439 5.9 1931.24

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14961	.260 %		
Amp y	.15000	.15005	-.032 %		
Freq x	.00540	.00540	.000000 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.3	.33	.003	.34
Phase y	70.0	70.2	.20	.056	.47
Pendulous data, amp, freq, phase			=:	.500	.03125 50.0
Equivalent U & V deflections (meters)			=:	59.0	58.8
iseed = -170000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =			593	1	593

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9436 5.9 1190.60
 1 POSITIVE 1.9436 5.9 1375.82
 2 POSITIVE 1.9436 5.9 1561.04
 3 POSITIVE 1.9436 5.9 1746.27
 4 POSITIVE 1.9436 5.9 1931.49

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MI degs
Amp x	.15000	.14971	.195 %		
Amp y	.15000	.14962	.255 %		
Freq x	.00540	.00540	.000002 Hz		
Freq y	.00540	.00540	.000000 Hz		
Phase x	160.0	160.4	.44	.148	1.16
Phase y	70.0	70.3	.33	.006	.35
Pendulous data, amp, freq, phase			=:	.500	.03125 50.0
Equivalent U & V deflections (meters)			=:	58.8	58.8
iseed = -180000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03

TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04

REV LABEL	POLARITY	RATE(D/S)	# OF ROTATIONS	START TIME
0	POSITIVE	1.9438	5.9	1190.46
1	POSITIVE	1.9438	5.9	1375.67
2	POSITIVE	1.9438	5.9	1560.87
3	POSITIVE	1.9438	5.9	1746.07
4	POSITIVE	1.9438	5.9	1931.27

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14941	.391 %		
Amp y	.15000	.15026	-.172 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000000 Hz		
Phase x	160.0	160.6	.55	.087	.98
Phase y	70.0	70.2	.23	.020	.33
Pendulous data, amp, freq, phase =:				.500	.03125 50.0
Equivalent U & V deflections (meters) =:				59.1	58.7
iseed = -190000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point =				593 1 593	

MIDPOINT TIME FOR THIS DATA WINDOW IS : .303104E+03
 TIME TAG ON LAST POINT IN BUFFER IS : .102298E+04
 REV LABEL POLARITY RATE(D/S) # OF ROTATIONS START TIME
 0 POSITIVE 1.9431 5.9 1190.84
 1 POSITIVE 1.9431 5.9 1376.11
 2 POSITIVE 1.9431 5.9 1561.38
 3 POSITIVE 1.9431 5.9 1746.64
 4 POSITIVE 1.9431 5.9 1931.91

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14966	.229 %		
Amp y	.15000	.14979	.139 %		
Freq x	.00540	.00540	.000002 Hz		
Freq y	.00540	.00540	.000003 Hz		
Phase x	160.0	160.5	.54	.149	1.26
Phase y	70.0	70.6	.58	.169	1.40
Pendulous data, amp, freq, phase =:				.500	.03125 50.0
Equivalent U & V deflections (meters) =:				58.9	58.8
iseed = -200000					
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14966	.226 %		
Amp y	.15000	.15003	-.017 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.4	.43	.074	.79
Phase y	70.0	70.3	.26	.069	.60
Pendulous data, amp, freq, phase =:				.500	.03125 50.0
Equivalent U & V deflections (meters) =:				59.0	58.8
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

	INPUT	OUTPUT	ERROR	PHASE GRAD degs/cycle	PHASE @ T=15 MIN degs
Amp x	.15000	.14965	.232 %		
Amp y	.15000	.15001	-.006 %		
Freq x	.00540	.00540	.000001 Hz		
Freq y	.00540	.00540	.000001 Hz		
Phase x	160.0	160.4	.41	.075	.77
Phase y	70.0	70.3	.30	.072	.64
Pendulous data, amp, freq, phase =:				.500	.03125 50.0
Equivalent U & V deflections (meters) =:				59.0	58.8
Noise data: 1 sigma value (deg/sec) :				.000280	
Stop index, start index, & number of point = 593				1 593	

APPENDIX 2.C

Programs for ECR Testing

- (1) SIMREAD - Read Simulation Data Files
- (2) SIMTEST - Compare Observer Output to Simulation Input
- (3) BTBUSO - Compare Observer Output to Model Signal Input

The program SIMREAD.FOR reads a simulation input file and creates the output file IDFTXY.DAT for the UNOMSC.FOR program (frequency domain skip-rope observer).

```

real x,y,tlength,time
integer amcsmode
open (10,file = 'TRUTH.DAT',status = 'old')
open (11,file = 'IDFTXY.DAT',status = 'unknown')
i = 1
1 read(10,*,end = 2) time, x, y, tlength, amcsmode, d1, d2
write(11,*) time, x, y, tlength, amcsmode
i = i + 1
go to 1
2 continue
lf = i - 1
print*, 'read ', lf, ' lines of data'
close(10,status = 'keep')
close(11,status = 'keep')
end

```

The program SIMTEST.FOR compares the output of UNOMSC.FOR to the original simulation input data file.

PROGRAM TO TEST FILTER AGAINST SIMULATION DATA

PHASE ANGLE AT TIME T IS DEFINED AS ARC TAN (V TERM/ U TERM)
 PHASE ERROR IS DEFINED AS 'TRUTH' - 'MODEL'.
 MAGNITUDE CALCULATED AS SQRT (U**2 + V**2)
 MAGNITUDE ERROR EXPRESSED AS PERCENT TERM BY

$$\text{ERROR} = 100\% * (\text{MAG}(\text{TRUTH}) - \text{MAG}(\text{MODEL})) / \text{MAG}(\text{TRUTH})$$

FINAL ERRORS ARE EXPRESSED AS RMS OVER ALL TIME POINTS

NAME IS 'SIMTEST'

INTEGER NDIM

REAL*4 DFR

PARAMETER (NDIM=4000, DFR=57.2958)

REAL*4 UM(NDIM), VM(NDIM), UT(NDIM), VT(NDIM)

REAL*4 RE(NDIM), RM(NDIM), RT(NDIM)

REAL*4 PE(NDIM), PM(NDIM), PT(NDIM)

OPEN INPUT FILES - 'F TXYUV.DAT' IS OUTPUT FROM 'UNOMSC.FOR'
 (MUST SET ODF TIME TO .TRUE. TO WRITE THIS FILE !!) AND 'TRUTH.DAT'
 IS THE SIMULATION DATA FILE (COPY THE APPROPRIATE SIMULATION DATA
 FILE INTO 'TRUTH.DAT' BEFORE RUNNING 'SIMTEST.FOR')

```

OPEN (11, FILE = 'F TXYUV.DAT', STATUS = 'OLD')
OPEN (12, FILE = 'TRUTH.DAT', STATUS = 'OLD')

```

READ THE INPUT FILES, AND CALCULATE THE TIME SAMPLING RATES AND
LENGTH OF EACH

READ (11, *, END=3) TIME1, D1, D2, UM(1), VM(1)
READ (11, *, END=3) TIME2, D1, D2, UM(2), VM(2)

I=3

READ (11, *, END=3) TIME, D1, D2, UM(I), VM(I)

I = I + 1

GO TO 2

LM = I - 1

DT_M = TIME2 - TIME1

READ (12, *, END=7) T0, D1, D2, D3, D4, UT(1), VT(1)
READ (12, *, END=7) T1, D1, D2, D3, D4, UT(2), VT(2)

I = 3

READ (12, *, END=7) T, D1, D2, D3, D4, UT(I), VT(I)

I = I + 1

GO TO 6

LT = I - 1

DT_T = T1 - T0

LC = MIN0 (LM, LT)

RLC = FLOAT (LC)

PRINT *, ' MODEL DATA HAS ', LM, ' TIME POINTS'

PRINT *, ' TRUTH DATA HAS ', LT, ' TIME POINTS'

PRINT *, ' WILL USE ', LC, ' TIME POINTS'

PRINT *, ' DT FOR MODEL DATA IS : ', DT_M

PRINT *, ' DT FOR TRUTH DATA IS : ', DT_T

WRITE(7,*) ' MODEL DATA HAS ', LM, ' TIME POINTS'

WRITE(7,*) ' MODEL DATA HAS ', LT, ' TIME POINTS'

WRITE(7,*) ' WILL USE ', LC, ' TIME POINTS'

WRITE(7,*) ' DT FOR MODEL DATA IS : ', DT_M

WRITE(7,*) ' DT FOR TRUTH DATA IS : ', DT_T

CALCULATE THE OVERALL RMS ERRORS IN THE MAGNITUDE AND PHASE

DO K=1, LC

RM(K)= SQRT(UM(K)**2 + VM(K)**2)

RT(K)= SQRT(UT(K)**2 + VT(K)**2)

PM(K)= DFR * ATAN2(VM(K), UM(K))

PT(K)= DFR * ATAN2(VT(K), UT(K))

RE(K)= 100. * (1. - RM(K)/RT(K))

END DO

CALL SATAN (PM, LC, 1)

CALL SATAN (PT, LC, 1)

SSQP = 0.

SSQR = 0.

DO K=1, LC

```

-      PE(K) = PM(K) - PT(K)
-      SSQP = SSQP + PE(K)**2
-      SSQR = SSQR + RE(K)**2
-    END DO

```

```

-    PHASE = SQRT ( SSQP/RLC )
-    RMAG = SQRT ( SSQR/RLC )
-    PRINT *, ' OVERALL RMS MAGNITUDE ERROR = ', RMAG, ' %'
-    PRINT *, ' OVERALL RMS PHASE ERROR = ', PHASE, ' DEGREES'

```

```

-    WRITE(7,*) ' OVERALL RMS MAGNITUDE ERROR = ', RMAG, ' %'
-    WRITE(7,*) ' OVERALL RMS PHASE ERROR = ', PHASE, ' DEGREES'

```

```

C    CLOSE FILES
-    CLOSE(11, STATUS = 'KEEP')
-    CLOSE(12, STATUS = 'KEEP')

```

```

-    END

```

```

-    SUBROUTINE SATAN (A, N, TYPE)

```

```

- C    SMART ATAN PROGRAM - REMOVES THE 2 PI JUMPS AT +- 180.
- C    AS WRITTEN THE PROGRAM IS A POST PROCESSOR. IT COULD BE
- C    ALTERED TO RUN ON-LINE.
- C    TYPE = 1 - ANGLES ARE IN DEGREES, BOTH INPUT AND OUTPUT.
- C    IF TYPE NE 1, THEN RADIANS WILL BE USED.
- C

```

```

- C    REAL*4 A(1), CHECK, ADD, PI
- C    INTEGER TYPE, N, JK, JK0
- C    LOGICAL FLAG
- C    PARAMETER (PI = 3.1415926)

```

```

- C    IF (TYPE .EQ. 1) THEN
- C        CHECK = 355.0
- C        ADD = 360.0
- C    ELSE
- C        CHECK = 6.19
- C        ADD = 2.0 * PI
- C    END IF

```

```

- C    JK0 = 2

```

```

- C    5    FLAG = .TRUE.
- C    JK = JK0

```

```

- C    DO K = JK, N
- C        IF (ABS(A(K) - A(K-1)) .GT. CHECK) THEN
- C            A(K) = A(K) + SIGN(ADD, A(K-1))
- C            IF (FLAG) THEN
- C                FLAG = .FALSE.
- C                JK0 = K

```

```

-         END IF
-     END IF
- END DO

```

```

C
- IF (.NOT. FLAG) GO TO 5
- RETURN
- END

```

The program VTBUSO.FOR compares the results of UNOMSC.FOR to the model signals generated by CREATE.FOR.

```

- open(3,file='F_REC.DAT',status='old')
- open(4,file='F_RECORD.DAT',status='old')
- open(7,file='T_REPORT.DAT',status='unknown')

```

```

- c read file from signal generating program (CREATE.FOR) output
- read(3,*) knoise, tensigma, iseed
- read(3,*) axin, fxin, pxin
- read(3,*) ayin, fyin, pyin
- read(3,*) apend, fpend, ppend

```

```

c make sure angles are bounded by pi.
- c subroutine ajsign puts angles in -pi to + pi range.
c

```

```

- pxin = ajsign (pxin)
- pyin = ajsign (pyin)

```

```

- frqin = 0.5 * (fxin + fyin)
- period = 1.0/frqin
- onesigma = tensigma/10.0

```

```

c read file from filter output
- read(4,*) tm
- read(4,*) lb, le, leb
- read(4,*) t0, tf, dt, tl, wk, psign
- read(4,*) ax, fx, px, ay, fy, py
- read(4,*) fa, ta, u, v, fl, fh
- read (4, *) avgx, avgy

```

```

- px = 57.296*px
- py = 57.296*py

```

```

- c
- epx = abs( abs( pxin ) - abs( px) )
- epy = abs( abs( pyin ) - abs( py) )
- if (axin .ne. 0.0) then
-     eax = 100.*(1. - ax/axin)
- else
-     eax = 357.
- endif

```

```

if (ayin .ne. 0.0) then
    eay = 100.*(1. - ay/ayin)
else
    eay = 357.
endif

pgx = 360.*abs(frqin - fx)*period
pgy = 360.*abs(frqin - fy)*period
phase_errx = epX + pgx * (900./period)
phase_erry = epy + pgy * (900./period)
close (3,status='keep')
close (4,status='keep')
write(7,9)
write(7,9)
write(7,9)
write(7,9)
write(7,10)
write(7,11) axin, ax, eax
write(7,12) ayin, ay, eay
write(7,9)
write(7,13) frqin, fx, abs(frqin-fx)
write(7,14) frqin, fy, abs(frqin-fy)
write(7,9)
write(7,15) pxin, px, epX, pgx, phase_errx
write(7,16) pyin, py, epy, pgy, phase_erry
write(7,9)
write(7,17) apend, fpend, ppend
write(7,18) u, v
if (knoise .eq. 1) then
    write(7,*) ' iseed = ', iseed
    write(7,20) onesigma
else
    write(7,*) ' No noise for this run.'
endif
write(7, 23) le, lb, leb

```

c
c

```

9  format(2x,' ')
10 format(12x,'INPUT',6x,'OUTPUT',8x,'ERROR',6x,'PHASE GRAD'
$,6x,'PHASE @ T=15 MIN',/48x,'deg/cycle',12x,'deg',/)
11 format(2x,'Amp x',f10.5,f12.5,f12.3,' %')
12 format(2x,'Amp y',f10.5,f12.5,f12.3,' %')
13 format(2x,'Freq x',f9.5,f12.5,f14.6,' Hz')
14 format(2x,'Freq y',f9.5,f12.5,f14.6,' Hz')
15 format(2x,'Phase x',f8.1,f12.1,f12.2,6x,f8.3,10x,f8.2)
16 format(2x,'Phase y',f8.1,f12.1,f12.2,6x,f8.3,10x,f8.2)
17 format(2x,'Pendulous data, amp, freq, phase      =:',f11.3
$, ,f11.5,f6.1)
18 format(2x,'Equivalent U & V deflections (meters) =:',2f9.1)
20 format(5x,'Noise data:      1 sigma value (deg/sec) :',f14.6)
22 format(2x,'Means in X and Y axes were :',2f12.7,' and'
$, ' were removed')

```

```
23  format(2x,'Stop index, start index, & number of point =',3i4)
```

```
write(7,9)  
write(7,9)
```

```
close (7, status='keep')  
call exit  
end
```

```
real function ajsign (x)  
if ( abs(x) .gt. 180. ) then  
    ajsign = x - sign(360., x)  
else  
    ajsign = x  
endif  
return  
end
```

APPENDIX 2.D

Programs for Systematic Testing

- (1) NO_NOISE - Noise-free Test Cases, Station 2 - page 1
- (2) NOISE - Noisy Test Cases, Station 2 - page 7
- (3) LIBRATION - Noise-free Tests at Station 1 - page 16
- (4) LIB_NOISE - Noisy Tests at Station 1 - page 23

-C Program NO_NOISE.FOR uses the model signal creation (without noise) algorithm
 C from the CREATE.FOR program and the WORK, HANN, MEAN, FOUR1, and LSCF sub-
 C routines from the UNOMSC.FOR program. NO_NOISE.FOR systematically runs
 -C through the phase pairings for a given frequency.

```

REAL X(3000)
COMMON/PAR1/LB,LE,NFT,NPNT,NDEG
COMMON/PAR2/DT,PI,DF,PID
dt = 1.024
sd = 2.8e-03
read*,f1x,iseed
lb=1
npnt = 7
ndeg = 3
le = int(1/(f1x*dt)+0.5)
if (le.eq.2*(le/2)) le = le+1
le = le*3 + lb - 1
NFT=8192
LEB=LE-LB+1
PI=4.0*ATAN(1.0)
DF=1.0/(NFT*DT)
PID=180.0/PI
do m = 1,37
  ph1x = (m-19)*10
  do n = 1,37
    ph2x = (n-19)*10
  CALL CREATE(x,A1X,F1X,PH1X,ph2x)
  CALL WORK(X,ampx,frx,phx,aerror,ferror,perror,A1X,F1X,PH1X)
  write(11,*)ph1x,ph2x,aerror
  write(12,*)ph1x,ph2x,ferror
  write(13,*)ph1x,ph2x,perror
  end do
end do
END

```

```

SUBROUTINE CREATE(xt,A1X,F1X,PH1X,ph2x)
REAL xt(3000)
dt = 1.024
iper=1000
a1x=0.02
a0x=0.065
a2x=0.5
PI = 4.0*ATAN(1.0)
convrt = pi/180.0
phi1x = ph1x*convrt
phi2x = ph2x*convrt
TPIDT = 2.0*PI*DT
F2X = 0.03125
xtheta = tpidt*f1x
xphi = tpidt*f2x
DO I = 1,IPER

```



```

-      I1 = I-1
-      xt(I) = A0X + A1X*COS(xtheta*I1+PHI1X)+A2X*COS(xphi*I1+PHI2X)
-      END DO
-      RETURN
-      END

```

```

- C      SUBROUTINE WORK CALCULATES THE AMPLITUDE, PHASE, AND FREQUENCY
- C      OF THE DATA. THE FOURIER TRANSFORM SUBROUTINE FOUR1 IS
- C      CALLED BY SUBROUTINE WORK. WORK RETURNS TO THE MAIN PROGRAM
- C      THE VALUES OF THE AMPLITUDE, PHASE, AND FREQUENCY AS WELL AS
- C      THE TIME INDEX WHERE THE MAXIMUM VALUE OCCURS.
- C      THIS IS BASED ON MODEL OF COS(WT+PHASE).
- C

```

```

-      SUBROUTINE WORK(ANG, amp, freq, phase, Aerror, Perror, Ferror, A1, F1, PHI1)

```

```

-      INTEGER NDIM, NCDIM
-      PARAMETER (NDIM=3000, NCDIM=8200, NFT=8192, NPNT=7)
-      DIMENSION AUX(NDIM), ANG(1)

```

```

-      REAL*4 XFREQ(7), PHIMAG(7), PHREAL(7)
-      COMPLEX AWO(NCDIM)

```

```

-      COMMON/PAR1/LB, LE
-      COMMON/PAR2/DT, PI, DF, PID

```

```

-      NTB1=LE-LB+1
-      NTB1 IS FORCED TO BE ODD IN MAIN PROGRAM.
-      HANN WINDOW ROUTINE USES ODD NUMBER OF POINTS TO TAPER.

```

```

- C      LOAD INPUT DATA FROM ANG(I) INTO ARRAY AUX(J).
- C      LB1=1-LB
- C      DO I=LB, LE
- C          IL=I+LB1
- C          AUX(IL)=ANG(I)
- C      END DO

```

```

- C      APPLY WINDOW FUNCTION TO TIME SEQUENCE
- C      CALL HANN (NTB1, AUX, BIAS)

```

```

- C      MAKE COMPLEX NUMBER AWO(I) FROM REAL NUMBER AUX(I) BY USING
- C      A ZERO IMAGINARY VALUE (AUX(I) IS THE REAL VALUE).
- C

```

```

-      DO I=1, NTB1
-          AWO(I)=CMPLX(AUX(I), 0.)
-      END DO

```

```

- C      NOW PAD THE DATA STREAM WITH ZEROS OUT TO AWO(8192).
- C

```

```

- C      DO I=NTB1+1,NFT
-          AWO(I) = CMPLX(0.,0.)
-      END DO

```

```

- C      SUBROUTINE FOUR1 DOES THE FOURIER TRANSFORM USING A FFT METHOD.
- C
- C      CALL FOUR1(AWO,NFT,1)

```

```

- C      LOOP TO FIND THE MAXIMUM MODULUS VALUE OF THE FOURIER TRANSFORM
-      ampMAX=0.0
-      istart = int((f1-0.001)/df)+1
-      iend = int((f1+0.001)/df)+1
-      do i = istart,iend
-          fr = (i-1)*df
-          if (cabs(awo(i)).gt.ampmax) then
-              ampmax = cabs(awo(i))
-              kf = i
-              freq = fr
-          end if
-      end do

```

```

- C      CREATE THE 3 DATA SETS TO BE FITTED BY LEAST SQUARES POLYNOMIAL.
- C      POLYNOMIAL IS 2ND DEGREE AND 7 POINTS WILL BE USED IN CURVE FIT.
- C      3 SETS ARE:
- C          MAGNITUDE OF TRANSFORM (SQRT(REAL**2 + IMAG**2))
- C          REAL PART
- C          IMAGINARY PART

```

```

- C      CENTER OF DATA SET IS THE FREQUENCY POINT WHERE MAX WAS FOUND.
- C
- C      DO I = 1,NPNT
-          J = KF-((NPNT+1)/2.0)+I
-          XFREQ(I) = CABS(AWO(J))
-          PHIMAG(I) = AIMAG(AWO(J))
-          PHREAL(I) = REAL(AWO(J))
-      END DO

```

```

- C      DO CURVE FIT ON THE MODULUS OF THE FOURIER TRANSFORM
- C      CALL TO LSCF WITH OPTION 1 DOES 2 THINGS.
- C      CURVE FITS AND COMPUTES TRUE MAX FREQUENCY POINT.

```

```

- C      CALL LSCF (FQ_P0, ampMAX, XFREQ, 1)

```

```

- C      CALL LSCF (FQ_P0, PHASEI, PHIMAG, 2)
- C      CALL LSCF (FQ_P0, PHASER, PHREAL, 2)
- C      FREQ = FREQ + DF * FQ_P0

```

```

- C      SCALING OF TRANSFORMED DATA IS PERFORMED TO GIVE OUTPUTS IN
- C

```

```

C DEGS/SEC AND REPRESENT ACTUAL RATE DATA.
C SCALE = 4.0/FLOAT(NTB1-1)
C AMP = SCALE * ampMAX
C PHASE = -ATAN2(PHASEI,PHASER)*pid

```

```

C THE FORWARD FOURIER TRANSFORM IS USUALLY DEFINED WITH EXP(-i*PI*F*T).
C MANY FFT ROUTINES, INCLUDING FOUR1, USE EXP(+i*2*PI*F*T). THESE TWO
C DIFFERENT CONVENTIONS FOR THE FORWARD FOURIER TRANSFORM RESULT IN TWO
C DIFFERENT FORMS FOR THE SHIFT THEOREM. IN THE FIRST CASE, THE SHIFT
C THEOREM STATES THAT IF G(T) TRANSFORMS AS G(F), THEN G(T+T1) TRANSFORMS
C AS EXP(i*2*PI*F*T1)*G(F). IN THE SECOND CASE, IF G(T) TRANSFORMS AS
C G(F), THEN G(T+T1) TRANSFORMS AS EXP(-i*2*PI*F*T1)*G(F). SINCE OUR
C MODEL IS COS(2*PI*F*T + P) = COS(2*PI*F*(T + P/(2*PI*F))), AND WE USE
C THE FIRST CONVENTION FOR THE FOURIER TRANSFORM, WE EXPECT OUR PHASE
C TO BE 2*PI*F*P/(2*PI*F) = P. HOWEVER, SINCE THE PROGRAM USES THE
C SECOND CONVENTION FOR THE FOURIER TRANSFORM, THE PHASE IS -P, SO TO
C CORRECT FOR THIS DIFFERENCE WE MUST INCLUDE ANOTHER - SIGN: -(-P) = P.

```

```

- ERROR = (ABS(FREQ-F1)/0.005) * 100
- AERROR = (ABS(AMP-A1)/A1) * 100
- PERROR = (ABS(PHASE-PH1)/360.0) * 100
- if (abs(phi1).eq.180) perror = (abs(abs(phase)-abs(phi1))/360.0)*100.0
- RETURN
- END

```

```

- SUBROUTINE HANN (LA,A11, BIAS)

```

```

C TAPER IS RAISED COSINE CURVE.
C MEAN IS COMPUTED AND REMOVED FROM INPUT SIGNAL
C PARAMETER (PI=3.1415926)
- REAL A11(LA), HW(3000)
- ITM = (LA-1)/2
- RM = FLOAT(ITM)
- DO IT= -ITM, ITM
- I = 1 + IT + ITM
- HW(I)=0.5*(1.0 + COS(PI*FLOAT(IT)/RM ) )
- A11(I)=A11(I)*HW(I)
- END DO
C COMPUTE MEAN OF TAPERED SIGNAL
C TRUE MEAN IS TWICE COMPUTED VALUE BECAUSE HANN WINDOW
C REDUCES VALUE BY FACTOR OF 2. (I.E. MEAN OF WINDOW IS 0.5)
- CALL MEAN (LA, A11, BIAS)
- BIAS = BIAS * 2.0 * LA/(LA-1)
- DO I = 1,LA
- A11(I) = A11(I) - BIAS * HW(I)
- END DO
- RETURN
- END

```

SUBROUTINE MEAN(LA,A22, SA)

THIS ROUTINE COMPUTES THE DC TERM OF THE DATA STREAM.
MEAN IS NOT REMOVED, BUT ONLY COMPUTED.

```
REAL A22(LA)
SA = 0.
DO I=1,LA
    SA=SA+A22(I)
END DO
SA=SA/FLOAT(LA)
RETURN
END
```

SUBROUTINE FOUR1(DATA,NN,ISIGN)

THIS ROUTINE DOES THE FOURIER TRANSFORM USING A FFT METHOD.

```
REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
DIMENSION DATA(*)
N=2*NN
J=1
DO 11 I=1,N,2
    IF(J.GT.I) THEN
        TEMPR=DATA(J)
        TEMPI=DATA(J+1)
        DATA(J)=DATA(I)
        DATA(J+1)=DATA(I+1)
        DATA(I)=TEMPR
        DATA(I+1)=TEMPI
    ENDIF
    M=N/2
    IF ((M.GE.2).AND.(J.GT.M)) THEN
        J=J-M
        M=M/2
        GO TO 1
    ENDIF
    J=J+M
11 CONTINUE
MMAX=2
IF (N.GT.MMAX) THEN
    ISTEP=2*MMAX
    THETA=6.28318530717959D0/(ISIGN*MMAX)
    WPR=-2.D0*DSIN(0.5D0*THETA)**2
    WPI=DSIN(THETA)
    WR=1.D0
    WI=0.D0
    DO 13 M=1,MMAX,2
        DO 12 I=M,N,ISTEP
            J=I+MMAX
            TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
```

```

-      TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
-      DATA(J)=DATA(I)-TEMPR
-      DATA(J+1)=DATA(I+1)-TEMPI
-      DATA(I)=DATA(I)+TEMPR
-      DATA(I+1)=DATA(I+1)+TEMPI
12    CONTINUE
      WTEMP=WR
      WR=WR*WPR-WI*WPI+WR
      WI=WI*WPR+WTEMP*WPI+WI
13    CONTINUE
      MMAX=ISTEP
-    GO TO 2
-    ENDIF
-    RETURN
-    END

```

```

- C    THIS SUBROUTINE DOES LEAST SQUARES CURVE FIT TO 7 POINTS
- C    FOR A 2ND DEGREE POLYNOMIAL. THE DATA IS ASSUMED TO BE SAMPLED
- C    AT INTEGRAL INTERVALS. ANY SCALING MUST BE DONE OUTSIDE
- C    THIS SUBROUTINE. THE 7 POINTS ARE :

```

```

- C    P = -3, -2, -1, 0, 1, 2, 3
- C    THE POLYNOMIAL IS  $F(P) = A + B*P + C*P*P$ .
- C    THE MAX OCCURS AT  $P = P_0 = -B/(2*C)$ .
- C    FREQUENCY CORRESPONDING TO  $P_0$  IS  $P_0*DF$  (DF OF DATA STREAM)
- C    THIS DELTA IS REFERENCED TO MIDPOINT FREQUENCY OF 7 POINTS.
- C    THE MAX VALUE IS  $F(P_0) = A - (B*B)/(4*C)$ .

```

```

- C    SUBROUTINE LSCF (P0, FMAX, U_IN, IOPT)

```

```

- C    ON ENTRY:

```

```

- C    U_IN IS INPUT ORDINATE VALUES. ( 7 )

```

```

- C    IOPT IS OPTION FOR 1 OF 2 THINGS

```

```

- C    1 : FIND P0 WHERE MAX OCCURS PLUS COMPUTE MAX VALUE.

```

```

- C    2 : COMPUTE VALUE OF POLYNOMIAL AT SPECIFIED FREQUENCY P0.

```

```

- C    IF IOPT=2, THEN P0 IS FREQUENCY POINT TO EVALUATE POLYNOMIAL.

```

```

- C    ON EXIT

```

```

- C    P0 IS VALUE OF P WHERE MAX PEAK OCCURS;

```

```

- C    THIS IS WRT CENTER POINT OF DATA.

```

```

- C    FMAX IS VALUE OF FUNCTION AT  $P=P_0$ .

```

```

- C    *****

```

```

- C    REAL*4 U_IN(*)

```

```

- C    LOGICAL G_FLAG

```

```

- C    *****

```

```

- C    FIRST STEP IS TO DO LEAST SQUARES.

```

```

- C    ALL COEFFICIENTS HAVE BEEN PRE-COMPUTED.

```

```

- C    'A' IS -8, 12, 24, 28, 24, 12, -8 DIVIDED BY 84

```

```

- C    'B' IS -9, -6, -3, 0, 3, 6, 9 DIVIDED BY 84

```

```

- C    'C' IS 5, 0, -3, -4, -3, 0, 5 DIVIDED BY 84

```

```

- C    US17 = U_IN(1) + U_IN(7)

```

```

US35 = U_IN(3) + U_IN(5)
- C A = COF1
COF1 = -8.*US17 + 12.*(U_IN(2)+U_IN(6)) +
# 24.*US35 + 28.*U_IN(4)
- C B = COF2
COF2 = 9.*(-U_IN(1)+U_IN(7) ) +
# 6.*(-U_IN(2)+U_IN(6) ) +
# 3.*(-U_IN(3)+U_IN(5) )
- C C= COF3
COF3 = 5.*US17 -3.*US35 - 4.*U_IN(4)
IF (ABS(COF3).LT.1.0E-08) THEN
PRINT*, '***** WARNING *****'
PRINT*, 'CONSTANT VALUE EQUALS ZERO - CANNOT COMPUTE A MAXIMUM'
PRINT*, 'FREQUENCY VALUE.'
RETURN
ENDIF
- C DID NOT DIVIDE BY 84 YET. DO SO FOR MAX PART BUT NOT P0.
- C COMPUTE P0, VALUE OF F(P) WHERE A+B*P+C*P*P = 0
- C COMPUTE FUNCTION AT P0; A+B*P0+C*P0*P0 = A-B**2/4C
- C IF (IOPT .EQ. 1) THEN
P0=-0.5*COF2/COF3
FMAX = (COF1 + 0.5 * P0 * COF2) /84.
- ELSE
FMAX = (COF1 + P0*( COF2 + P0*COF3 ) )/84.
- END IF
- RETURN
- END

```

C Program NOISE.FOR uses the signal creation and noise generation algorithms
C from the CREATE.FOR program and the WORK, HANN, MEAN, FOUR1, and LSCF sub-
C routines from the UNOMSC.FOR program. NOISE.FOR systematically runs through
C the phase pairings for a given frequency.

```

- REAL X(3000),xinit(3000),ax(100),fx(100),px(100),axe(100),fxe(100),
# pxe(100)
EXTERNAL ran1
- COMMON/PAR1/LB,LE,NFT,NPNT,NDEG
COMMON/PAR2/DT,PI,DF,PID
dt = 1.024
sd = 2.8e-03
- read*,flx,iseed
lb=1
npnt = 7
- ndeg = 3
le = int(1/(flx*dt)+0.5)
if (le.eq.2*(le/2)) le = le+1
- le = le*3 + lb - 1

```

```

      NFT=8192
      LEB=LE-LB+1
      PI=4.0*ATAN(1.0)
      DF=1.0/(NFT*DT)
      PID=180.0/PI
      do m = 1,37
        ph1x = (m-19)*10
        do n = 1,37
          ph2x = (n-19)*10
          sumax = 0.0
          sumfx = 0.0
          sumpx = 0.0
          sumaxe = 0.0
          sumfxe = 0.0
          sumpxe = 0.0
          amax = 0.0
          fmax = 0.0
          pmax = 0.0
          CALL CREATE(xinit,A1X,F1X,PH1X,ph2x)
c Loop to create noise in data
          do k = 1,50
            DO I = 1,1000,2
              v1 = 2.0*ran1(idum) - 1.0
              v2 = 2.0*ran1(idum) - 1.0
              r = v1**2 + v2**2
              if (r.ge.1) go to 1
              fac = sqrt(-2.0*log(r)/r)*sd
              x(i) = v1*fac + xinit(i)
              x(i+1) = v2*fac + xinit(i+1)
            END DO
            CALL WORK(X,ampx,frx,phx,AXer,PXer,FXer,A1X,F1X,PH1X)
            ax(k) = ampx
            fx(k) = frx
            px(k) = phx
            axe(k) = axer
            fxe(k) = fxer
            pxe(k) = pxer
            if (axer.gt.amax) amax = axer
            if (fxer.gt.fmax) fmax = fxer
            if (pxer.gt.pmax) pmax = pxer
            sumax = sumax + ampx
            sumfx = sumfx + frx
            if (abs(ph1x).eq.180) then
              sumpx = sumpx + abs(phx)
            else
              sumpx = sumpx + phx
            end if
            sumaxe = sumaxe + axer
            sumfxe = sumfxe + fxer
            sumpxe = sumpxe + pxer
          end do
          avgax = sumax/50.0

```

```

avgfx = sumfx/50.0
avgpx = sumpx/50.0
avgaxe = sumaxe/50.0
avgfxe = sumfxe/50.0
avgpxe = sumpxe/50.0
sumax2 = 0.0
sumfx2 = 0.0
sumpx2 = 0.0
sumaxe2 = 0.0
sumfxe2 = 0.0
sumpxe2 = 0.0
do j = 1,50
    sumax2 = sumax2 + (ax(j) - avgax)**2
    sumfx2 = sumfx2 + (fx(j) - avgfx)**2
    if (abs(ph1x).eq.180) then
        sumpx2 = sumpx2 + (abs(px(j)) - avgpx)**2
    else
        sumpx2 = sumpx2 + (px(j) - avgpx)**2
    end if
    sumaxe2 = sumaxe2 + (axe(j) - avgaxe)**2
    sumfxe2 = sumfxe2 + (fxe(j) - avgfxe)**2
    sumpxe2 = sumpxe2 + (pxe(j) - avgpxe)**2
end do
sdax = sqrt(sumax2/49.0)
sdfx = sqrt(sumfx2/49.0)
sdpx = sqrt(sumpx2/49.0)
sdaxe = sqrt(sumaxe2/49.0)
sdfxe = sqrt(sumfxe2/49.0)
sdpxe = sqrt(sumpxe2/49.0)
write(8,*)ph1x,ph2x,amax
write(9,*)ph1x,ph2x,fmax
write(10,*)ph1x,ph2x,pmax
write(95,*)ph1x,ph2x,avgaxe
write(96,*)ph1x,ph2x,avgfxe
write(97,*)ph1x,ph2x,avgpxe
end do
end do
END

```

```

SUBROUTINE CREATE(xt,A1X,F1X,PH1X,ph2x)
REAL xt(3000)
dt = 1.024
iper=1000
a1x=0.02
a0x=0.065
a2x=0.5
PI = 4.0*ATAN(1.0)
convrt = pi/180.0
ph1x = ph1x*convrt
ph2x = ph2x*convrt
TPIDT = 2.0*PI*DT
F2X = 0.03125

```



```

-
- xtheta = tpidt*f1x
- xphi = tpidt*f2x
- DO I = 1,IPER
- I1 = I-1
-   xt(I) = A0X + A1X*COS(xtheta*I1+PHI1X)+A2X*COS(xphi*I1+PHI2X)
- END DO
- RETURN
- END

```

```

- C   SUBROUTINE WORK CALCULATES THE AMPLITUDE, PHASE, AND FREQUENCY
- C   OF THE DATA. THE FOURIER TRANSFORM SUBROUTINE FOUR1 IS
- C   CALLED BY SUBROUTINE WORK. WORK RETURNS TO THE MAIN PROGRAM
- C   THE VALUES OF THE AMPLITUDE, PHASE, AND FREQUENCY AS WELL AS
- C   THE TIME INDEX WHERE THE MAXIMUM VALUE OCCURS.
- C   THIS IS BASED ON MODEL OF COS(WT+PHASE).
- C

```

```

- SUBROUTINE WORK(ANG,amp,freq,phase,Aerror,Perror,Ferror,A1,F1,PHI1)

```

```

-   INTEGER NDIM, NCDIM
-   PARAMETER (NDIM=3000, NCDIM=8200, NFT=8192, NPNT=7)
-   DIMENSION AUX(NDIM), ANG(1)

```

```

-   REAL*4 XFREQ(7), PHIMAG(7), PHREAL(7)
-   COMPLEX AWO(NCDIM)

```

```

-   COMMON/PAR1/LB,LE
-   COMMON/PAR2/DT,PI,DF,PID

```

```

-   NTB1=LE-LB+1
-   NTB1 IS FORCED TO BE ODD IN MAIN PROGRAM.
-   HANN WINDOW ROUTINE USES ODD NUMBER OF POINTS TO TAPER.

```

```

- C   LOAD INPUT DATA FROM ANG(I) INTO ARRAY AUX(J).
- C   LB1=1-LB
- C   DO I=LB,LE
- C       IL=I+LB1
- C       AUX(IL)=ANG(I)
-   END DO

```

```

- C   APPLY WINDOW FUNCTION TO TIME SEQUENCE
- C   CALL HANN (NTB1,AUX,BIAS)

```

```

- C   MAKE COMPLEX NUMBER AWO(I) FROM REAL NUMBER AUX(I) BY USING
- C   A ZERO IMAGINARY VALUE (AUX(I) IS THE REAL VALUE).
- C

```

```

-   DO I=1,NTB1
-       AWO(I)=CMPLX(AUX(I),0.)
-   END DO

```

```
C
C NOW PAD THE DATA STREAM WITH ZEROS OUT TO AWO(8192).
C
```

```
- DO I=NTB1+1,NFT
  AWO(I) = CMPLX(0.,0.)
END DO
```

```
- C
C SUBROUTINE FOUR1 DOES THE FOURIER TRANSFORM USING A FFT METHOD.
C
```

```
- CALL FOUR1(AWO,NFT,1)
```

```
C LOOP TO FIND THE MAXIMUM MODULUS VALUE OF THE FOURIER TRANSFORM
- ampMAX=0.0
  istart = int((f1-0.001)/df)+1
  iend = int((f1+0.001)/df)+1
  do i = istart,iend
    fr = (i-1)*df
    if (cabs(awo(i)).gt.ampmax) then
      ampmax = cabs(awo(i))
      kf = i
      freq = fr
    end if
  end do
```

```
- C
C CREATE THE 3 DATA SETS TO BE FITTED BY LEAST SQUARES POLYNOMIAL.
C POLYNOMIAL IS 2ND DEGREE AND 7 POINTS WILL BE USED IN CURVE FIT.
```

```
C 3 SETS ARE:
C MAGNITUDE OF TRANSFORM (SQRT(REAL**2 + IMAG**2))
C REAL PART
C IMAGINARY PART
```

```
- C
C CENTER OF DATA SET IS THE FREQUENCY POINT WHERE MAX WAS FOUND.
C
```

```
- DO I = 1,NPNT
  J = KF-((NPNT+1)/2.0)+I
  XFREQ(I) = CABS(AWO(J))
  PHIMAG(I) = AIMAG(AWO(J))
  PHREAL(I) = REAL(AWO(J))
END DO
```

```
- C
C DO CURVE FIT ON THE MODULUS OF THE FOURIER TRANSFORM
C CALL TO LSCF WITH OPTION 1 DOES 2 THINGS.
C CURVE FITS AND COMPUTES TRUE MAX FREQUENCY POINT.
```

```
- CALL LSCF (FQ_P0, ampMAX, XFREQ, 1)
```

```
- C
CALL LSCF (FQ_P0, PHASEI, PHIMAG, 2)
CALL LSCF (FQ_P0, PHASER, PHREAL, 2)
```

```
FREQ = FREQ + DF * FQ_PO
```

```
SCALING OF TRANSFORMED DATA IS PERFORMED TO GIVE OUTPUTS IN  
DEGS/SEC AND REPRESENT ACTUAL RATE DATA.
```

```
SCALE = 4.0/FLOAT(NTB1-1)
```

```
AMP = SCALE * ampMAX
```

```
PHASE = -ATAN2(PHASEI, PHASER)*pid
```

```
THE FORWARD FOURIER TRANSFORM IS USUALLY DEFINED WITH  $\exp(-i\pi F T)$ .  
MANY FFT ROUTINES, INCLUDING FOUR1, USE  $\exp(+i2\pi F T)$ . THESE TWO  
DIFFERENT CONVENTIONS FOR THE FORWARD FOURIER TRANSFORM RESULT IN TWO  
DIFFERENT FORMS FOR THE SHIFT THEOREM. IN THE FIRST CASE, THE SHIFT  
THEOREM STATES THAT IF  $G(T)$  TRANSFORMS AS  $G(F)$ , THEN  $G(T+T1)$  TRANSFORMS  
AS  $\exp(i2\pi F T1) * G(F)$ . IN THE SECOND CASE, IF  $G(T)$  TRANSFORMS AS  
 $G(F)$ , THEN  $G(T+T1)$  TRANSFORMS AS  $\exp(-i2\pi F T1) * G(F)$ . SINCE OUR  
MODEL IS  $\cos(2\pi F T + P) = \cos(2\pi F (T + P/(2\pi F)))$ , AND WE USE  
THE FIRST CONVENTION FOR THE FOURIER TRANSFORM, WE EXPECT OUR PHASE  
TO BE  $2\pi F P / (2\pi F) = P$ . HOWEVER, SINCE THE PROGRAM USES THE  
SECOND CONVENTION FOR THE FOURIER TRANSFORM, THE PHASE IS  $-P$ , SO TO  
CORRECT FOR THIS DIFFERENCE WE MUST INCLUDE ANOTHER - SIGN:  $-(-P) = P$ .
```

```
ERROR = (ABS(FREQ-F1)/0.005) * 100
```

```
AERROR = (ABS(AMP-A1)/A1) * 100
```

```
PERROR = (ABS(PHASE-PH1)/360.0) * 100
```

```
if (abs(phi1).eq.180) perror = (abs(abs(phase)-abs(phi1))/360.0)*100.0
```

```
RETURN
```

```
END
```

```
SUBROUTINE HANN (LA,A11, BIAS)
```

```
TAPER IS RAISED COSINE CURVE.
```

```
MEAN IS COMPUTED AND REMOVED FROM INPUT SIGNAL
```

```
PARAMETER (PI=3.1415926)
```

```
REAL A11(LA), HW(3000)
```

```
ITM = (LA-1)/2
```

```
RM = FLOAT(ITM)
```

```
DO IT= -ITM, ITM
```

```
  I = 1 + IT + ITM
```

```
  HW(I)=0.5*(1.0 + COS(PI*FLOAT(IT)/RM) )
```

```
  A11(I)=A11(I)*HW(I)
```

```
END DO
```

```
COMPUTE MEAN OF TAPERED SIGNAL
```

```
TRUE MEAN IS TWICE COMPUTED VALUE BECAUSE HANN WINDOW  
REDUCES VALUE BY FACTOR OF 2. (I.E. MEAN OF WINDOW IS 0.5)
```

```
CALL MEAN (LA, A11, BIAS)
```

```
BIAS = BIAS * 2.0 * LA/(LA-1)
```

```
DO I = 1,LA
```

```
  A11(I) = A11(I) - BIAS * HW(I)
```

```
END DO
```

```
RETURN
```

```
END
```

SUBROUTINE MEAN(LA,A22, SA)

THIS ROUTINE COMPUTES THE DC TERM OF THE DATA STREAM.
MEAN IS NOT REMOVED, BUT ONLY COMPUTED.

```
REAL A22(LA)
SA = 0.
DO I=1,LA
  SA=SA+A22(I)
END DO
SA=SA/FLOAT(LA)
RETURN
END
```

SUBROUTINE FOUR1(DATA,NN,ISIGN)

THIS ROUTINE DOES THE FOURIER TRANSFORM USING A FFT METHOD.

```
REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
DIMENSION DATA(*)
N=2*NN
J=1
DO 11 I=1,N,2
  IF(J.GT.I)THEN
    TEMPR=DATA(J)
    TEMPI=DATA(J+1)
    DATA(J)=DATA(I)
    DATA(J+1)=DATA(I+1)
    DATA(I)=TEMPR
    DATA(I+1)=TEMPI
  ENDIF
  M=N/2
  IF ((M.GE.2).AND.(J.GT.M)) THEN
    J=J-M
    M=M/2
    GO TO 1
  ENDIF
  J=J+M
11 CONTINUE
MMAX=2
2 IF (N.GT.MMAX) THEN
  ISTEP=2*MMAX
  THETA=6.28318530717959D0/(ISIGN*MMAX)
  WPR=-2.D0*DSIN(0.5D0*THETA)**2
  WPI=DSIN(THETA)
  WR=1.D0
  WI=0.D0
  DO 13 M=1,MMAX,2
    DO 12 I=M,N,ISTEP
      J=I+MMAX
```

```

-      TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
-      TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
-      DATA(J)=DATA(I)-TEMPR
-      DATA(J+1)=DATA(I+1)-TEMPI
-      DATA(I)=DATA(I)+TEMPR
-      DATA(I+1)=DATA(I+1)+TEMPI
12    CONTINUE
-      WTEMP=WR
-      WR=WR*WPR-WI*WPI+WR
-      WI=WI*WPR+WTEMP*WPI+WI
13    CONTINUE
-      MMAX=ISTEP
-      GO TO 2
-      ENDIF
-      RETURN
-      END

```

```

- C      THIS SUBROUTINE DOES LEAST SQUARES CURVE FIT TO 7 POINTS
- C      FOR A 2ND DEGREE POLYNOMIAL. THE DATA IS ASSUMED TO BE SAMPLED
- C      AT INTEGRAL INTERVALS. ANY SCALING MUST BE DONE OUTSIDE
- C      THIS SUBROUTINE. THE 7 POINTS ARE :
- C      P = -3, -2, -1, 0, 1, 2, 3
- C      THE POLYNOMIAL IS  $F(P) = A + B*P + C*P*P$ .
- C      THE MAX OCCURS AT  $P = P_0 = -B/(2*C)$ .
- C      FREQUENCY CORRESPONDING TO  $P_0$  IS  $P_0*DF$  (DF OF DATA STREAM)
- C      THIS DELTA IS REFERENCED TO MIDPOINT FREQUENCY OF 7 POINTS.
- C      THE MAX VALUE IS  $F(P_0) = A - (B*B)/(4*C)$ .
- C
- C      SUBROUTINE LSCF (P0, FMAX, U_IN, IOPT)
- C
- C      ON ENTRY:
- C      U_IN IS INPUT ORDINATE VALUES. ( 7 )
- C      IOPT IS OPTION FOR 1 OF 2 THINGS
- C      1 : FIND P0 WHERE MAX OCCURS PLUS COMPUTE MAX VALUE.
- C      2 : COMPUTE VALUE OF POLYNOMIAL AT SPECIFIED FREQUENCY P0.
- C      IF IOPT=2, THEN P0 IS FREQUENCY POINT TO EVALUATE POLYNOMIAL.
- C
- C      ON EXIT
- C      P0 IS VALUE OF P WHERE MAX PEAK OCCURS;
- C      THIS IS WRT CENTER POINT OF DATA.
- C      FMAX IS VALUE OF FUNCTION AT P=P0.
- C      *****
- C      REAL*4 U_IN(*)
- C      LOGICAL G_FLAG
- C      *****
- C      FIRST STEP IS TO DO LEAST SQUARES.
- C      ALL COEFFICIENTS HAVE BEEN PRE-COMPUTED.
- C      'A' IS -8, 12, 24, 28, 24, 12, -8 DIVIDED BY 84
- C      'B' IS -9, -6, -3, 0, 3, 6, 9 DIVIDED BY 84
- C      'C' IS 5, 0, -3, -4, -3, 0, 5 DIVIDED BY 84
- C

```

```

US17 = U_IN(1) + U_IN(7)
US35 = U_IN(3) + U_IN(5)
C A = COF1
COF1 = -8.*US17 + 12.*(U_IN(2)+U_IN(6)) +
# 24.*US35 + 28.*U_IN(4)
C B = COF2
COF2 = 9.*(-U_IN(1)+U_IN(7) ) +
# 6.*(-U_IN(2)+U_IN(6) ) +
# 3.*(-U_IN(3)+U_IN(5) )
C C= COF3
COF3 = 5.*US17 -3.*US35 - 4.*U_IN(4)
IF (ABS(COF3).LT.1.0E-08) THEN
PRINT*, '***** WARNING *****'
PRINT*, 'CONSTANT VALUE EQUALS ZERO - CANNOT COMPUTE A MAXIMUM'
PRINT*, 'FREQUENCY VALUE.'
RETURN
ENDIF

C DID NOT DIVIDE BY 84 YET. DO SO FOR MAX PART BUT NOT P0.
C
C COMPUTE P0, VALUE OF F(P) WHERE A+B*P+C*P*P = 0
C COMPUTE FUNCTION AT P0; A+B*P0+C*P0*P0 = A-B**2/4C
C IF (IOPT .EQ. 1) THEN
P0=-0.5*COF2/COF3
FMAX = (COF1 + 0.5 * P0 * COF2) /84.

ELSE
FMAX = (COF1 + P0*( COF2 + P0*COF3 ) )/84.
END IF
RETURN
END

function ran1(idum)
dimension r(97)
parameter (m1 = 259200, ia1 = 7141, ic1 = 54773, rm1 = 1.0/m1)
parameter (m2 = 134456, ia2 = 8121, ic2 = 28411, rm2 = 1.0/m2)
parameter (m3 = 243000, ia3 = 4561, ic3 = 51349)
data iff /0/
if (idum.lt.0.or.iff.eq.0) then
iff = 1
ix1 = mod(ic1 - idum,m1)
ix1 = mod(ia1*ix1 + ic1,m1)
ix2 = mod(ix1,m2)
ix1 = mod(ia1*ix1 + ic1,m1)
ix3 = mod(ix1,m3)
do j = 1,97
ix1 = mod(ia1*ix1 + ic1,m1)
ix2 = mod(ia2*ix2 + ic2,m2)
r(j) = (float(ix1) + float(ix2)*rm2)*rm1
end do
idum = 1
end if

```

```

ix1 = mod(ia1*ix1 + ic1,m1)
ix2 = mod(ia2*ix2 + ic2,m2)
ix3 = mod(ia3*ix3 + ic3,m3)
j = 1 + (97*ix3)/m3
if (j.gt.97.or.j.lt.1) pause
ran1 = r(j)
r(j) = (float(ix1) + float(ix2)*rm2)*rm1
return
end

```

C Program LIBRATION.FOR uses the signal (no noise) generation algorithms from
 C the CRELIBR.FOR program and the WORK, HANN, MEAN, FOUR1, and LSCF subroutines
 C from the UNOMSC.FOR program. LIBRATION.FOR systematically runs through the
 C phase pairings for a given frequency.

```

REAL X(4000)
COMMON/PAR1/LB,LE,NFT,NPNT,NDEG
COMMON/PAR2/DT,PI,DF,PID
dt = 1.024
sd = 2.8e-03
lb=1
npnt = 7
ndeg = 3
f1x = 0.0019
read*,numcycle,iseed
le = int(1/(f1x*dt)+0.5)
if (le.eq.2*(le/2)) le = le+1
le = le*numcycle + lb - 1
NFT=8192
LEB=LE-LB+1
PI=4.0*ATAN(1.0)
DF=1.0/(NFT*DT)
PID=180.0/PI
do m = 1,37
  ph1x = (m-19)*10
  do n = 1,37
    ph2x = (n-19)*10
  CALL CREATE(x,A1X,F1X,PH1X,ph2x)
  CALL WORK(X,ampx,frx,phx,aerror,ferror,perror,A1X,F1X,PH1X)
  write(11,*)ph1x,ph2x,aerror
  write(12,*)ph1x,ph2x,ferror
  write(13,*)ph1x,ph2x,perror
  end do
end do
END

```

```

SUBROUTINE CREATE(x,A1X,F1X,PHI1X,phi2x)
REAL x(4000)
tlength=20000.0
dt = 1.024
iper=3000
a1x=0.0034
a0x=0.065
a2x=0.05
alibx = 0.0046
PI = 4.0*ATAN(1.0)
convrt = pi/180.0
ph1x = phi1x*convrt
ph2x = phi2x*convrt
TPIDT = 2.0*PI*DT
f1x = 0.0019
F2X = 0.089
flibx = 1/2713.0
xtheta = tpidt*f1x
xphi = tpidt*f2x
xlibr = tpidt*flibx
DO I = 1,IPER
    I1 = I-1
    TIM =I1*DT
    x(I) = A0X + A1X*COS(xtheta*I1+PH1X)+A2X*COS(xphi*I1) +
$      alibx*cos(xlibr*i1+ph2x)
END DO
RETURN
END

```

SUBROUTINE WORK CALCULATES THE AMPLITUDE, PHASE, AND FREQUENCY
 OF THE DATA. THE FOURIER TRANSFORM SUBROUTINE FOUR1 IS
 CALLED BY SUBROUTINE WORK. WORK RETURNS TO THE MAIN PROGRAM
 THE VALUES OF THE AMPLITUDE, PHASE, AND FREQUENCY AS WELL AS
 THE TIME INDEX WHERE THE MAXIMUM VALUE OCCURS.
 THIS IS BASED ON MODEL OF $\cos(WT + \text{PHASE})$.

```

SUBROUTINE WORK(ANG,amp,freq,phase,Aerror,Perror,Ferror,A1,F1,PHI1)

```

```

INTEGER NDIM, NCDIM
PARAMETER (NDIM=4000, NCDIM=8200, NFT=8192, NPNT=7).
DIMENSION AUX(NDIM),ANG(1)

```

```

REAL*4 XFREQ(7), PHIMAG(7),PHREAL(7)
COMPLEX AWO(NCDIM)

```

```

COMMON/PAR1/LB,LE
COMMON/PAR2/DT,PI,DF,PID

```

```

NTB1=LE-LB+1

```

NTB1 IS FORCED TO BE ODD IN MAIN PROGRAM.
 HANN WINDOW ROUTINE USES ODD NUMBER OF POINTS TO TAPER.


```

C
C
C  LOAD INPUT DATA FROM ANG(I) INTO ARRAY AUX(J).

```

```

LB1=1-LB
DO I=LB,LE
    IL=I+LB1
    AUX(IL)=ANG(I)
END DO

```

```

C
C  APPLY WINDOW FUNCTION TO TIME SEQUENCE
CALL HANN (NTB1,AUX,BIAS)

```

```

C
C  MAKE COMPLEX NUMBER AWO(I) FROM REAL NUMBER AUX(I) BY USING
C  A ZERO IMAGINARY VALUE (AUX(I) IS THE REAL VALUE).
C

```

```

DO I=1,NTB1
    AWO(I)=CMPLX(AUX(I),0.)
END DO

```

```

C
C  NOW PAD THE DATA STREAM WITH ZEROS OUT TO AWO(8192)..
C

```

```

DO I=NTB1+1,NFT
    AWO(I) = CMPLX(0.,0.)
END DO

```

```

C
C  SUBROUTINE FOUR1 DOES THE FOURIER TRANSFORM USING A FFT METHOD.
C

```

```

CALL FOUR1(AWO,NFT,1)

```

```

C  LOOP TO FIND THE MAXIMUM MODULUS VALUE OF THE FOURIER TRANSFORM
ampMAX=0.0

```

```

istart = int((f1-0.001)/df)+1
iend = int((f1+0.001)/df)+1
do i = istart,iend
    fr = (i-1)*df
    if (cabs(awo(i)).gt.ampmax) then
        ampmax = cabs(awo(i))
        kf = i
        freq = fr
    end if
end do

```

```

- C CREATE THE 3 DATA SETS TO BE FITTED BY LEAST SQUARES POLYNOMIAL.
- C POLYNOMIAL IS 2ND DEGREE AND 7 POINTS WILL BE USED IN CURVE FIT.
  C

```

```

  C 3 SETS ARE:
  C

```

```

  C MAGNITUDE OF TRANSFORM (SQRT(REAL**2 + IMAG**2))
- C REAL PART
  C IMAGINARY PART
  C

```

```

- C CENTER OF DATA SET IS THE FREQUENCY POINT WHERE MAX WAS FOUND.
  C

```

```

DO I = 1, NPNT
  J = KF - ((NPNT+1)/2.0) + I
  XFREQ(I) = CABS(AWO(J))
  PHIMAG(I) = AIMAG(AWO(J))
  PHREAL(I) = REAL(AWO(J))
END DO
  C

```

```

- C DO CURVE FIT ON THE MODULUS OF THE FOURIER TRANSFORM
- C CALL TO LSCF WITH OPTION 1 DOES 2 THINGS.
  C CURVE FITS AND COMPUTES TRUE MAX FREQUENCY POINT.
  C

```

```

- C CALL LSCF (FQ_P0, ampMAX, XFREQ, 1)
  C

```

```

- C CALL LSCF (FQ_P0, PHASEI, PHIMAG, 2)
- C CALL LSCF (FQ_P0, PHASER, PHREAL, 2)
  C FREQ = FREQ + DF * FQ_P0
  C

```

```

- C SCALING OF TRANSFORMED DATA IS PERFORMED TO GIVE OUTPUTS IN
- C DEGS/SEC AND REPRESENT ACTUAL RATE DATA.
  C SCALE = 4.0/FLOAT(NTB1-1)
- C AMP = SCALE * ampMAX
  C PHASE = -ATAN2(PHASEI, PHASER)*pid
  C

```

```

  C THE FORWARD FOURIER TRANSFORM IS USUALLY DEFINED WITH  $\exp(-i\pi F T)$ .
  C MANY FFT ROUTINES, INCLUDING FOUR1, USE  $\exp(+i2\pi F T)$ . THESE TWO
- C DIFFERENT CONVENTIONS FOR THE FORWARD FOURIER TRANSFORM RESULT IN TWO
  C DIFFERENT FORMS FOR THE SHIFT THEOREM. IN THE FIRST CASE, THE SHIFT
  C THEOREM STATES THAT IF  $G(T)$  TRANSFORMS AS  $G(F)$ , THEN  $G(T+T1)$  TRANSFORMS
- C AS  $\exp(i2\pi F T1) * G(F)$ . IN THE SECOND CASE, IF  $G(T)$  TRANSFORMS AS
  C  $G(F)$ , THEN  $G(T+T1)$  TRANSFORMS AS  $\exp(-i2\pi F T1) * G(F)$ . SINCE OUR
  C MODEL IS  $\cos(2\pi F T + P) = \cos(2\pi F (T + P/(2\pi F)))$ , AND WE USE
- C THE FIRST CONVENTION FOR THE FOURIER TRANSFORM, WE EXPECT OUR PHASE
  C TO BE  $2\pi F P / (2\pi F) = P$ . HOWEVER, SINCE THE PROGRAM USES THE
  C SECOND CONVENTION FOR THE FOURIER TRANSFORM, THE PHASE IS  $-P$ , SO TO
- C CORRECT FOR THIS DIFFERENCE WE MUST INCLUDE ANOTHER - SIGN:  $-(-P) = P$ .
  C

```

```

- C FERROR = (ABS(FREQ-F1)/0.005) * 100
- C AERROR = (ABS(AMP-A1)/A1) * 100
- C perror = (abs(abs(phase)-abs(phi1))/360.0)*100.0
  C RETURN
  C END
  C

```

SUBROUTINE HANN (LA,A11, BIAS)

TAPER IS RAISED COSINE CURVE.

MEAN IS COMPUTED AND REMOVED FROM INPUT SIGNAL

PARAMETER (PI=3.1415926)

REAL A11(LA), HW(3000)

ITM = (LA-1)/2

RM = FLOAT(ITM)

DO IT= -ITM, ITM

I = 1 + IT + ITM

HW(I)=0.5*(1.0 + COS(PI*FLOAT(IT)/RM))

A11(I)=A11(I)*HW(I)

END DO

COMPUTE MEAN OF TAPERED SIGNAL

TRUE MEAN IS TWICE COMPUTED VALUE BECAUSE HANN WINDOW

REDUCES VALUE BY FACTOR OF 2. (I.E. MEAN OF WINDOW IS 0.5)

CALL MEAN (LA, A11, BIAS)

BIAS = BIAS * 2.0 * LA/(LA-1)

DO I = 1,LA

A11(I) = A11(I) - BIAS * HW(I)

END DO

RETURN

END

SUBROUTINE MEAN(LA,A22, SA)

THIS ROUTINE COMPUTES THE DC TERM OF THE DATA STREAM.
MEAN IS NOT REMOVED, BUT ONLY COMPUTED.

REAL A22(LA)

SA = 0.

DO I=1,LA

SA=SA+A22(I)

END DO

SA=SA/FLOAT(LA)

RETURN

END

SUBROUTINE FOUR1(DATA,NN,ISIGN)

THIS ROUTINE DOES THE FOURIER TRANSFORM USING A FFT METHOD.

REAL*8 WR,WI,WPR,WPI,WTEMP,THETA

DIMENSION DATA(*)

N=2*NN

J=1

DO 11 I=1,N,2

IF(J.GT.I) THEN

TEMPR=DATA(J)

```

-      TEMPI=DATA(J+1)
-      DATA(J)=DATA(I)
-      DATA(J+1)=DATA(I+1)
-      DATA(I)=TEMPR
-      DATA(I+1)=TEMPI
-    ENDIF
-    M=N/2
- 1  IF ((M.GE.2).AND.(J.GT.M)) THEN
-      J=J-M
-      M=M/2
-      GO TO 1
-    ENDIF
-      J=J+M
- 11 CONTINUE
- 2  MMAX=2
- 2  IF (N.GT.MMAX) THEN
-      ISTEP=2*MMAX
-      THETA=6.28318530717959D0/(ISIGN*MMAX)
-      WPR=-2.D0*DSIN(0.5D0*THETA)**2
-      WPI=DSIN(THETA)
-      WR=1.D0
-      WI=0.D0
-      DO 13 M=1,MMAX,2
-          DO 12 I=M,N,ISTEP
-              J=I+MMAX
-              TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
-              TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
-              DATA(J)=DATA(I)-TEMPR
-              DATA(J+1)=DATA(I+1)-TEMPI
-              DATA(I)=DATA(I)+TEMPR
-              DATA(I+1)=DATA(I+1)+TEMPI
- 12          CONTINUE
-              WTEMP=WR
-              WR=WR*WPR-WI*WPI+WR
-              WI=WI*WPR+WTEMP*WPI+WI
- 13          CONTINUE
-              MMAX=ISTEP
-          GO TO 2
-      ENDIF
-      RETURN
-      END

```

C THIS SUBROUTINE DOES LEAST SQUARES CURVE FIT TO 7 POINTS
 C FOR A 2ND DEGREE POLYNOMIAL. THE DATA IS ASSUMED TO BE SAMPLED
 C AT INTEGRAL INTERVALS. ANY SCALING MUST BE DONE OUTSIDE
 C THIS SUBROUTINE. THE 7 POINTS ARE :
 C $P = -3, -2, -1, 0, 1, 2, 3$
 C THE POLYNOMIAL IS $F(P) = A + B*P + C*P*P$.
 C THE MAX OCCURS AT $P = P_0 = -B/(2*C)$.
 C FREQUENCY CORRESPONDING TO P_0 IS P_0*DF (DF OF DATA STREAM)
 C THIS DELTA IS REFERENCED TO MIDPOINT FREQUENCY OF 7 POINTS.

THE MAX VALUE IS $F(P_0) = A - (B*B)/(4*C)$.

SUBROUTINE LSCF (P0, FMAX, U_IN, IOPT)

ON ENTRY:

U_IN IS INPUT ORDINATE VALUES. (7)

IOPT IS OPTION FOR 1 OF 2 THINGS

1 : FIND P0 WHERE MAX OCCURS PLUS COMPUTE MAX VALUE.

2 : COMPUTE VALUE OF POLYNOMIAL AT SPECIFIED FREQUENCY P0.

IF IOPT=2, THEN P0 IS FREQUENCY POINT TO EVALUATE POLYNOMIAL.

ON EXIT

P0 IS VALUE OF P WHERE MAX PEAK OCCURS;

THIS IS WRT CENTER POINT OF DATA.

FMAX IS VALUE OF FUNCTION AT P=P0.

REAL*4 U_IN(*)

LOGICAL G_FLAG

FIRST STEP IS TO DO LEAST SQUARES.

ALL COEFFICIENTS HAVE BEEN PRE-COMPUTED.

'A' IS -8, 12, 24, 28, 24, 12, -8 DIVIDED BY 84

'B' IS -9, -6, -3, 0, 3, 6, 9 DIVIDED BY 84

'C' IS 5, 0, -3, -4, -3, 0, 5 DIVIDED BY 84

US17 = U_IN(1) + U_IN(7)

US35 = U_IN(3) + U_IN(5)

A = COF1

COF1 = -8.*US17 + 12.*(U_IN(2)+U_IN(6)) +
24.*US35 + 28.*U_IN(4)

B = COF2

COF2 = 9.*(-U_IN(1)+U_IN(7)) +
6.*(-U_IN(2)+U_IN(6)) +
3.*(-U_IN(3)+U_IN(5))

C= COF3

COF3 = 5.*US17 -3.*US35 - 4.*U_IN(4)

IF (ABS(COF3).LT.1.0E-08) THEN

PRINT*, '***** WARNING *****'

PRINT*, 'CONSTANT VALUE EQUALS ZERO - CANNOT COMPUTE A MAXIMUM'

PRINT*, 'FREQUENCY VALUE.'

RETURN

ENDIF

DID NOT DIVIDE BY 84 YET. DO SO FOR MAX PART BUT NOT P0.

COMPUTE P0, VALUE OF F(P) WHERE $A+B*P+C*P*P = 0$

COMPUTE FUNCTION AT P0; $A+B*P_0+C*P_0*P_0 = A-B**2/4C$

IF (IOPT .EQ. 1) THEN

P0=-0.5*COF2/COF3

FMAX = (COF1 + 0.5 * P0 * COF2) /84.

ELSE


```

DO I = 1,1000,2
  v1 = 2.0*ran1(idum) - 1.0
  v2 = 2.0*ran1(idum) - 1.0
  r = v1**2 + v2**2
  if (r.ge.1) go to 1
  fac = sqrt(-2.0*log(r)/r)*sd
  x(i) = v1*fac + xinit(i)
  x(i+1) = v2*fac + xinit(i+1)
END DO
CALL WORK(X,ampx,frx,phx,AXer,PXer,FXer,A1X,F1X,PH1X)
ax(k) = ampx
fx(k) = frx
px(k) = phx
axe(k) = axer
fxe(k) = fxer
pxe(k) = pxer
if (axer.gt.amax) amax = axer
if (fxer.gt.fmax) fmax = fxer
if (pxer.gt.pmax) pmax = pxer
sumax = sumax + ampx
sumfx = sumfx + frx
if (abs(ph1x).eq.180) then
  sumpx = sumpx + abs(phx)
else
  sumpx = sumpx + phx
end if
sumaxe = sumaxe + axer
sumfxe = sumfxe + fxer
sumpxe = sumpxe + pxer
end do
avgax = sumax/50.0
avgfx = sumfx/50.0
avgpx = sumpx/50.0
avgaxe = sumaxe/50.0
avgfxe = sumfxe/50.0
avgpxe = sumpxe/50.0
sumax2 = 0.0
sumfx2 = 0.0
sumpx2 = 0.0
sumaxe2 = 0.0
sumfxe2 = 0.0
sumpxe2 = 0.0
do j = 1,50
  sumax2 = sumax2 + (ax(j) - avgax)**2
  sumfx2 = sumfx2 + (fx(j) - avgfx)**2
  if (abs(ph1x).eq.180) then
    sumpx2 = sumpx2 + (abs(px(j)) - avgpx)**2
  else
    sumpx2 = sumpx2 + (px(j) - avgpx)**2
  end if
  sumaxe2 = sumaxe2 + (axe(j) - avgaxe)**2
  sumfxe2 = sumfxe2 + (fxe(j) - avgfxe)**2

```

```

-
-      sumpxe2 = sumpxe2 + (pxe(j) - avgpxe)**2
- end do
- sdax = sqrt(sumax2/49.0)
- sdfx = sqrt(sumfx2/49.0)
- sdpdx = sqrt(sumpdx2/49.0)
- sdaxe = sqrt(sumaxe2/49.0)
- sdfxe = sqrt(sumfxe2/49.0)
- sdpxe = sqrt(sumpxe2/49.0)
- write(8,*)ph1x,ph2x,amax
- write(9,*)ph1x,ph2x,fmax
- write(10,*)ph1x,ph2x,pmax
- write(95,*)ph1x,ph2x,avgaxe
- write(96,*)ph1x,ph2x,avgfxe
- write(97,*)ph1x,ph2x,avgpxe
- end do
- end do
- END

```

```

- SUBROUTINE CREATE(x,A1X,F1X,PHI1X,phi2x)

```

```

- REAL x(3000),y(3000)
- tlength=20000.0

```

```

- dt = 1.024

```

```

- iper=3000

```

```

- alx=0.0034

```

```

- a0x=0.065

```

```

- a2x=0.05

```

```

- alibx = 0.0046

```

```

- PI = 4.0*ATAN(1.0)

```

```

- convrt = pi/180.0

```

```

- ph1x = phi1x*convrt

```

```

- ph2x = phi2x*convrt

```

```

- TPIDT = 2.0*PI*DT

```

```

- f1x = 0.0019

```

```

- F2X = 0.089

```

```

- flibx = 1/2713.0

```

```

- xtheta = tpidt*f1x

```

```

- xphi = tpidt*f2x

```

```

- xlibr = tpidt*flibx

```

```

- DO I = 1,IPER

```

```

-   I1 = I-1

```

```

-   TIM =I1*DT

```

```

-   x(I) = A0X + A1X*COS(xtheta*I1+PH1X)+A2X*COS(xphi*I1) +

```

```

-   $   alibx*cos(xlibr*i1+ph2x)

```

```

- END DO

```

```

- RETURN

```

```

- END

```


SUBROUTINE WORK CALCULATES THE AMPLITUDE, PHASE, AND FREQUENCY
 OF THE DATA. THE FOURIER TRANSFORM SUBROUTINE FOUR1 IS
 CALLED BY SUBROUTINE WORK. WORK RETURNS TO THE MAIN PROGRAM
 THE VALUES OF THE AMPLITUDE, PHASE, AND FREQUENCY AS WELL AS
 THE TIME INDEX WHERE THE MAXIMUM VALUE OCCURS.
 THIS IS BASED ON MODEL OF $\cos(WT + \text{PHASE})$.

SUBROUTINE WORK(ANG, amp, freq, phase, Aerror, Perror, Ferror, A1, F1, PHI1)

INTEGER NDIM, NCDIM
 PARAMETER (NDIM=3000, NCDIM=8200, NFT=8192, NPNT=7)
 DIMENSION AUX(NDIM), ANG(1)

REAL*4 XFREQ(7), PHIMAG(7), PHREAL(7)
 COMPLEX AWO(NCDIM)

COMMON/PAR1/LB, LE
 COMMON/PAR2/DT, PI, DF, PID

NTB1=LE-LB+1
 NTB1 IS FORCED TO BE ODD IN MAIN PROGRAM.
 HANN WINDOW ROUTINE USES ODD NUMBER OF POINTS TO TAPER.

LOAD INPUT DATA FROM ANG(I) INTO ARRAY AUX(J).
 LB1=1-LB
 DO I=LB, LE
 IL=I+LB1
 AUX(IL)=ANG(I)
 END DO

APPLY WINDOW FUNCTION TO TIME SEQUENCE
 CALL HANN (NTB1, AUX, BIAS)

MAKE COMPLEX NUMBER AWO(I) FROM REAL NUMBER AUX(I) BY USING
 A ZERO IMAGINARY VALUE (AUX(I) IS THE REAL VALUE).

DO I=1, NTB1
 AWO(I)=CMPLX(AUX(I), 0.)
 END DO

NOW PAD THE DATA STREAM WITH ZEROS OUT TO AWO(8192).

DO I=NTB1+1, NFT
 AWO(I) = CMPLX(0., 0.)
 END DO

```

C
C SUBROUTINE FOUR1 DOES THE FOURIER TRANSFORM USING A FFT METHOD.
C

```

```

CALL FOUR1(AWO,NFT,1)

```

```

C LOOP TO FIND THE MAXIMUM MODULUS VALUE OF THE FOURIER TRANSFORM

```

```

ampMAX=0.0

```

```

istart = int((f1-0.001)/df)+1

```

```

iend = int((f1+0.001)/df)+1

```

```

do i = istart,iend

```

```

    fr = (i-1)*df

```

```

    if (cabs(awo(i)).gt.ampmax) then

```

```

        ampmax = cabs(awo(i))

```

```

        kf = i

```

```

        freq = fr

```

```

    end if

```

```

end do

```

```

C
C CREATE THE 3 DATA SETS TO BE FITTED BY LEAST SQUARES POLYNOMIAL.
C POLYNOMIAL IS 2ND DEGREE AND 7 POINTS WILL BE USED IN CURVE FIT.

```

```

    3 SETS ARE:

```

```

        MAGNITUDE OF TRANSFORM (SQRT(REAL**2 + IMAG**2))

```

```

        REAL PART

```

```

        IMAGINARY PART

```

```

C
C CENTER OF DATA SET IS THE FREQUENCY POINT WHERE MAX WAS FOUND.
C

```

```

DO I = 1,NPNT

```

```

    J = KF-((NPNT+1)/2.0)+I

```

```

    XFREQ(I) = CABS(AWO(J))

```

```

    PHIMAG(I) = AIMAG(AWO(J))

```

```

    PHREAL(I) = REAL(AWO(J))

```

```

END DO

```

```

C
C DO CURVE FIT ON THE MODULUS OF THE FOURIER TRANSFORM
C CALL TO LSCF WITH OPTION 1 DOES 2 THINGS.

```

```

    CURVE FITS AND COMPUTES TRUE MAX FREQUENCY POINT.

```

```

CALL LSCF (FQ_P0, ampMAX, XFREQ, 1)

```

```

CALL LSCF (FQ_P0, PHASEI, PHIMAG, 2)

```

```

CALL LSCF (FQ_P0, PHASER, PHREAL, 2)

```

```

FREQ = FREQ + DF * FQ_P0

```

```

C
C SCALING OF TRANSFORMED DATA IS PERFORMED TO GIVE OUTPUTS IN
C DEGS/SEC AND REPRESENT ACTUAL RATE DATA.

```

```

SCALE = 4.0/FLOAT(NBT1-1)

```

```

AMP = SCALE * ampMAX

```

```

PHASE = -ATAN2(PHASEI,PHASER)*pid

```

THE FORWARD FOURIER TRANSFORM IS USUALLY DEFINED WITH $\text{EXP}(-i*PI*F*T)$. MANY FFT ROUTINES, INCLUDING FOUR1, USE $\text{EXP}(+i*2*PI*F*T)$. THESE TWO DIFFERENT CONVENTIONS FOR THE FORWARD FOURIER TRANSFORM RESULT IN TWO DIFFERENT FORMS FOR THE SHIFT THEOREM. IN THE FIRST CASE, THE SHIFT THEOREM STATES THAT IF $G(T)$ TRANSFORMS AS $G(F)$, THEN $G(T+T1)$ TRANSFORMS AS $\text{EXP}(i*2*PI*F*T1)*G(F)$. IN THE SECOND CASE, IF $G(T)$ TRANSFORMS AS $G(F)$, THEN $G(T+T1)$ TRANSFORMS AS $\text{EXP}(-i*2*PI*F*T1)*G(F)$. SINCE OUR MODEL IS $\text{COS}(2*PI*F*T + P) = \text{COS}(2*PI*F*(T + P/(2*PI*F)))$, AND WE USE THE FIRST CONVENTION FOR THE FOURIER TRANSFORM, WE EXPECT OUR PHASE TO BE $2*PI*F*P/(2*PI*F) = P$. HOWEVER, SINCE THE PROGRAM USES THE SECOND CONVENTION FOR THE FOURIER TRANSFORM, THE PHASE IS $-P$, SO TO CORRECT FOR THIS DIFFERENCE WE MUST INCLUDE ANOTHER - SIGN: $-(-P) = P$.

```

FERROR = (ABS(FREQ-F1)/0.005) * 100
AERROR = (ABS(AMP-A1)/A1) * 100
perror = (abs(abs(phase)-abs(phi1))/360.0)*100.0
RETURN
END

```

SUBROUTINE HANN (LA,A11, BIAS)

TAPER IS RAISED COSINE CURVE.

MEAN IS COMPUTED AND REMOVED FROM INPUT SIGNAL

PARAMETER (PI=3.1415926)

REAL A11(LA), HW(3000)

ITM = (LA-1)/2

RM = FLOAT(ITM)

DO IT= -ITM, ITM

I = 1 + IT + ITM

HW(I)=0.5*(1.0 + COS(PI*FLOAT(IT)/RM))

A11(I)=A11(I)*HW(I)

END DO

COMPUTE MEAN OF TAPERED SIGNAL

TRUE MEAN IS TWICE COMPUTED VALUE BECAUSE HANN WINDOW

REDUCES VALUE BY FACTOR OF 2. (I.E. MEAN OF WINDOW IS 0.5)

CALL MEAN (LA, A11, BIAS)

BIAS = BIAS * 2.0 * LA/(LA-1)

DO I = 1,LA

A11(I) = A11(I) - BIAS * HW(I)

END DO

RETURN

END

SUBROUTINE MEAN(LA,A22, SA)

THIS ROUTINE COMPUTES THE DC TERM OF THE DATA STREAM.
MEAN IS NOT REMOVED, BUT ONLY COMPUTED.

REAL A22(LA)
SA = 0.
DO I=1,LA
 SA=SA+A22(I)
END DO
SA=SA/FLOAT(LA)
RETURN
END

SUBROUTINE FOUR1(DATA,NN,ISIGN)

THIS ROUTINE DOES THE FOURIER TRANSFORM USING A FFT METHOD.

REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
DIMENSION DATA(*)
N=2*NN
J=1
DO 11 I=1,N,2
 IF(J.GT.I) THEN
 TEMPR=DATA(J)
 TEMPI=DATA(J+1)
 DATA(J)=DATA(I)
 DATA(J+1)=DATA(I+1)
 DATA(I)=TEMPR
 DATA(I+1)=TEMPI
 ENDIF
 M=N/2
 1 IF ((M.GE.2).AND.(J.GT.M)) THEN
 J=J-M
 M=M/2
 GO TO 1
 ENDIF
 J=J+M
11 CONTINUE
MMAX=2
2 IF (N.GT.MMAX) THEN
 ISTEP=2*MMAX
 THETA=6.28318530717959D0/(ISIGN*MMAX)
 WPR=-2.D0*DSIN(0.5D0*THETA)**2
 WPI=DSIN(THETA)
 WR=1.D0
 WI=0.D0
 DO 13 M=1,MMAX,2
 DO 12 I=M,N,ISTEP
 J=I+MMAX
 TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)

```

      TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
      DATA(J)=DATA(I)-TEMPR
      DATA(J+1)=DATA(I+1)-TEMPI
      DATA(I)=DATA(I)+TEMPR
      DATA(I+1)=DATA(I+1)+TEMPI
12      CONTINUE
      WTEMP=WR
      WR=WR*WPR-WI*WPI+WR
      WI=WI*WPR+WTEMP*WPI+WI
13      CONTINUE
      MMAX=ISTEP
GO TO 2
ENDIF
RETURN
END

```

```

C      THIS SUBROUTINE DOES LEAST SQUARES CURVE FIT TO 7 POINTS
C      FOR A 2ND DEGREE POLYNOMIAL. THE DATA IS ASSUMED TO BE SAMPLED
C      AT INTEGRAL INTERVALS. ANY SCALING MUST BE DONE OUTSIDE
C      THIS SUBROUTINE. THE 7 POINTS ARE :
C      P = -3, -2, -1, 0, 1, 2, 3
C      THE POLYNOMIAL IS  $F(P) = A + B \cdot P + C \cdot P^2$ .
C      THE MAX OCCURS AT  $P = P_0 = -B/(2 \cdot C)$ .
C      FREQUENCY CORRESPONDING TO  $P_0$  IS  $P_0 \cdot DF$  (DF OF DATA STREAM)
C      THIS DELTA IS REFERENCED TO MIDPOINT FREQUENCY OF 7 POINTS.
C      THE MAX VALUE IS  $F(P_0) = A - (B \cdot B)/(4 \cdot C)$ .

```

```

SUBROUTINE LSCF (P0, FMAX, U_IN, IOPT)

```

```

ON ENTRY:

```

```

U_IN IS INPUT ORDINATE VALUES. (7 )

```

```

IOPT IS OPTION FOR 1 OF 2 THINGS

```

```

1 : FIND P0 WHERE MAX OCCURS PLUS COMPUTE MAX VALUE.

```

```

2 : COMPUTE VALUE OF POLYNOMIAL AT SPECIFIED FREQUENCY P0.

```

```

IF IOPT=2, THEN P0 IS FREQUENCY POINT TO EVALUATE POLYNOMIAL.

```

```

ON EXIT

```

```

P0 IS VALUE OF P WHERE MAX PEAK OCCURS;

```

```

THIS IS WRT CENTER POINT OF DATA.

```

```

FMAX IS VALUE OF FUNCTION AT  $P=P_0$ .

```

```

*****

```

```

REAL*4 U_IN(*)

```

```

LOGICAL G_FLAG

```

```

*****

```

```

FIRST STEP IS TO DO LEAST SQUARES.

```

```

ALL COEFFICIENTS HAVE BEEN PRE-COMPUTED.

```

```

'A' IS -8, 12, 24, 28, 24, 12, -8 DIVIDED BY 84

```

```

'B' IS -9, -6, -3, 0, 3, 6, 9 DIVIDED BY 84

```

```

'C' IS 5, 0, -3, -4, -3, 0, 5 DIVIDED BY 84

```

```

US17 = U_IN(1) + U_IN(7)

```

```

US35 = U_IN(3) + U_IN(5)
C A = COF1
COF1 = -8.*US17 + 12.*(U_IN(2)+U_IN(6)) +
# 24.*US35 + 28.*U_IN(4)
C B = COF2
COF2 = 9.*(-U_IN(1)+U_IN(7) ) +
# 6.*(-U_IN(2)+U_IN(6) ) +
# 3.*(-U_IN(3)+U_IN(5) )
C C= COF3
COF3 = 5.*US17 -3.*US35 - 4.*U_IN(4)
IF (ABS(COF3).LT.1.0E-08) THEN
PRINT*, '***** WARNING *****'
PRINT*, 'CONSTANT VALUE EQUALS ZERO - CANNOT COMPUTE A MAXIMUM'
PRINT*, 'FREQUENCY VALUE.'
RETURN
ENDIF
C DID NOT DIVIDE BY 84 YET. DO SO FOR MAX PART BUT NOT P0.
C
C COMPUTE P0, VALUE OF F(P) WHERE A+B*P+C*P*P = 0
C COMPUTE FUNCTION AT P0; A+B*P0+C*P0*P0 = A-B**2/4C
C IF (IOPT .EQ. 1) THEN
P0=-0.5*COF2/COF3
FMAX = (COF1 + 0.5 * P0 * COF2) /84.

ELSE
FMAX = (COF1 + P0*( COF2 + P0*COF3 ) )/84.
END IF
RETURN
END

function ran1(idum)
dimension r(97)
parameter (m1 = 259200, ia1 = 7141, ic1 = 54773, rm1 = 1.0/m1)
parameter (m2 = 134456, ia2 = 8121, ic2 = 28411, rm2 = 1.0/m2)
parameter (m3 = 243000, ia3 = 4561, ic3 = 51349)
data iff /0/
if (idum.lt.0.or.iff.eq.0) then
iff = 1
ix1 = mod(ic1 - idum,m1)
ix1 = mod(ia1*ix1 + ic1,m1)
ix2 = mod(ix1,m2)
ix1 = mod(ia1*ix1 + ic1,m1)
ix3 = mod(ix1,m3)
do j = 1,97
ix1 = mod(ia1*ix1 + ic1,m1)
ix2 = mod(ia2*ix2 + ic2,m2)
r(j) = (float(ix1) + float(ix2)*rm2)*rm1
end do
idum = 1
end if
ix1 = mod(ia1*ix1 + ic1,m1)

```

```
ix2 = mod(ia2*ix2 + ic2,m2)
ix3 = mod(ia3*ix3 + ic3,m3)
j = 1 + (97*ix3)/m3
if (j.gt.97.or.j.lt.1) pause
ran1 = r(j)
r(j) = (float(ix1) + float(ix2)*rm2)*rm1
return
end
```

APPENDIX 2.E

Simulation Test Results

These are the results of simulation 3:

MODEL DATA HAS	591	TIME POINTS	
MODEL DATA HAS	901	TIME POINTS	
WILL USE	591	TIME POINTS	
DT FOR MODEL DATA IS :	1.000000000		
DT FOR TRUTH DATA IS :	1.000000000		
OVERALL RMS MAGNITUDE ERROR =	34.42346191		%
OVERALL RMS PHASE ERROR =	22.10563469		DEGREES

The results of simulation 4 are as follows:

MODEL DATA HAS	1601	TIME POINTS	
MODEL DATA HAS	1801	TIME POINTS	
WILL USE	1601	TIME POINTS	
DT FOR MODEL DATA IS :	1.000000000		
DT FOR TRUTH DATA IS :	1.000000000		
OVERALL RMS MAGNITUDE ERROR =	9.227395058		%
OVERALL RMS PHASE ERROR =	6.498259544		DEGREES

These are the results of simulation 5:

MODEL DATA HAS	1601	TIME POINTS	
MODEL DATA HAS	1801	TIME POINTS	
WILL USE	1601	TIME POINTS	
DT FOR MODEL DATA IS :	1.000000000		
DT FOR TRUTH DATA IS :	1.000000000		
OVERALL RMS MAGNITUDE ERROR =	5.796232224		%
OVERALL RMS PHASE ERROR =	4.697713375		DEGREES

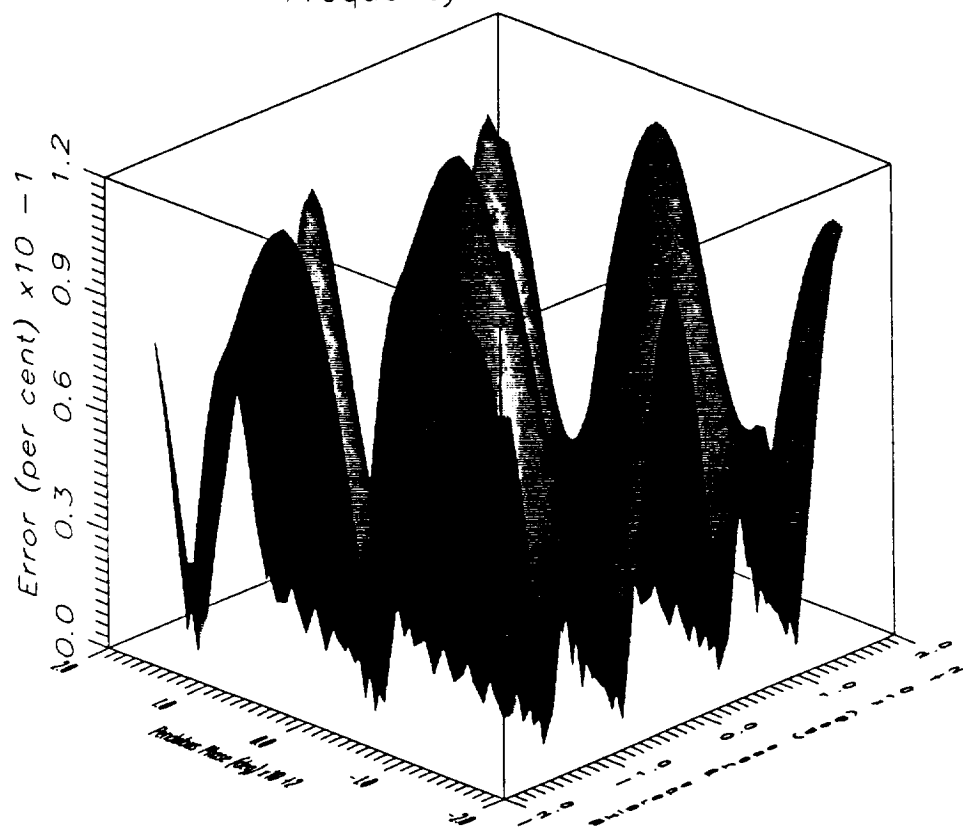
APPENDIX 2.F

Systematic Test Results

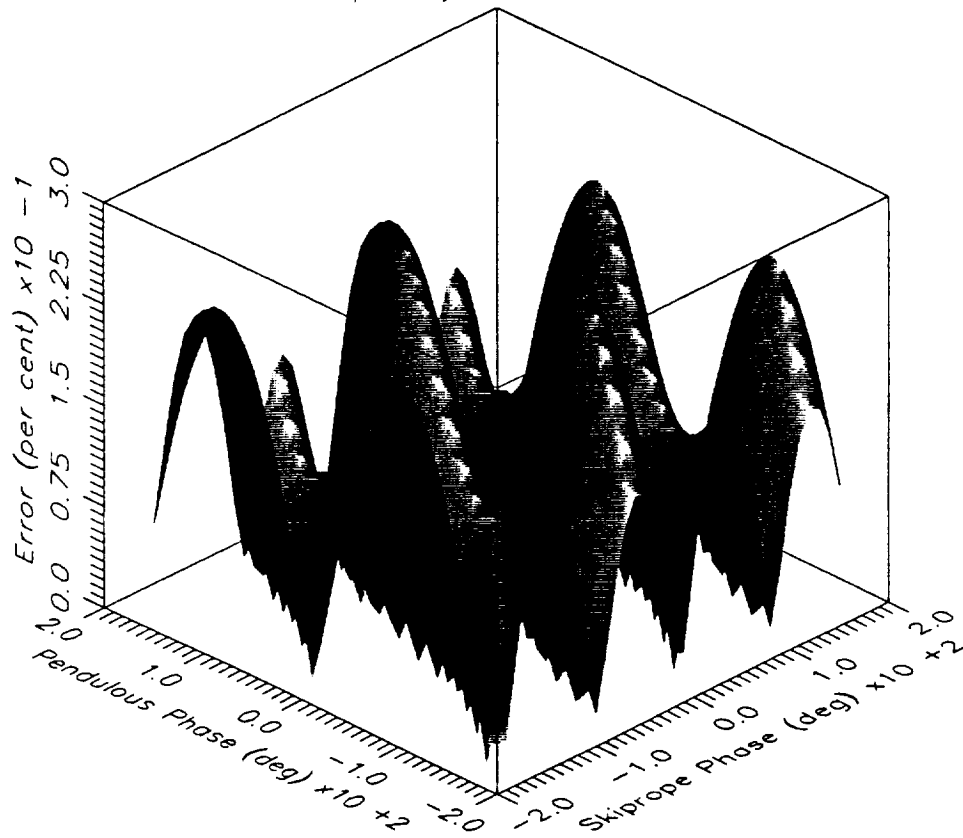
- (1) NO_NOISE - Noise-free Test Cases, Station 2
- (2) NOISE - Noisy Test Cases, Station 2
- (3) LIBRATION - Noise-free Tests at Station 1
- (4) LIB_NOISE - Noisy Tests at Station 1

The following 33 plots are the results of the NO_NOISE.FOR program. (Refer to section I part C of the test plan.) These plots represent the amplitude, frequency, and phase errors for the eleven skiprope frequencies running from 0.0045 Hz to 0.0055 Hz vs. the penduluos and skiprope phases.

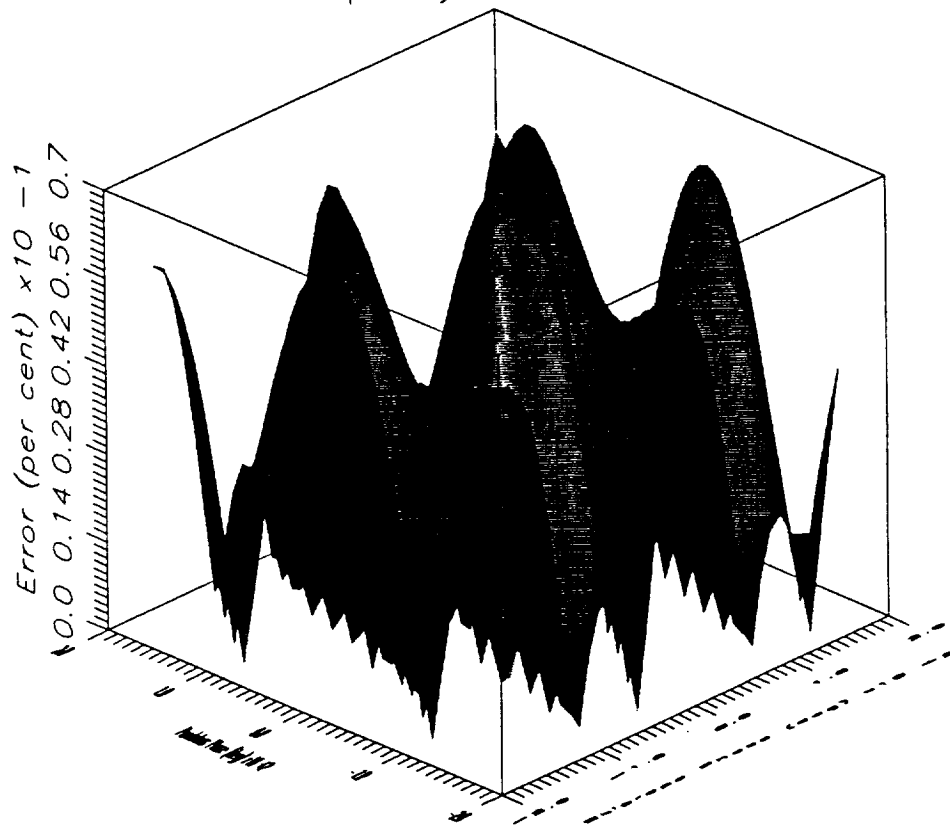
Maximum Amplitude Error = 0.118%
Frequency = 0.0045 Hz



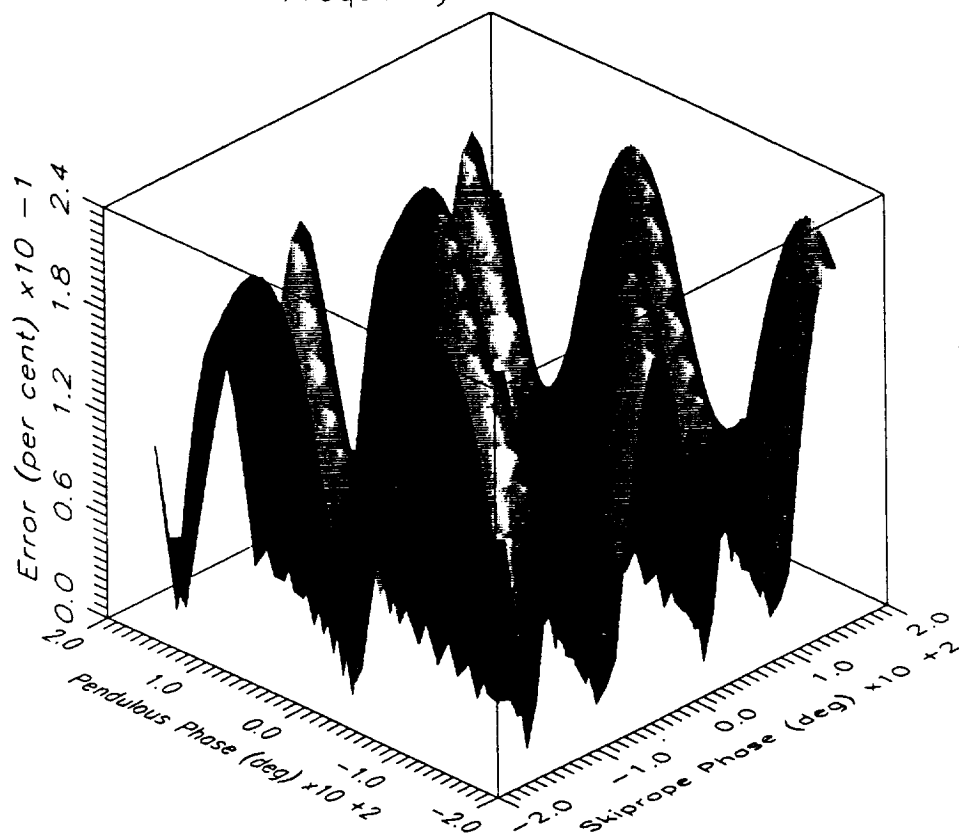
Maximum Amplitude Error = 0.253%
Frequency = 0.0048 Hz



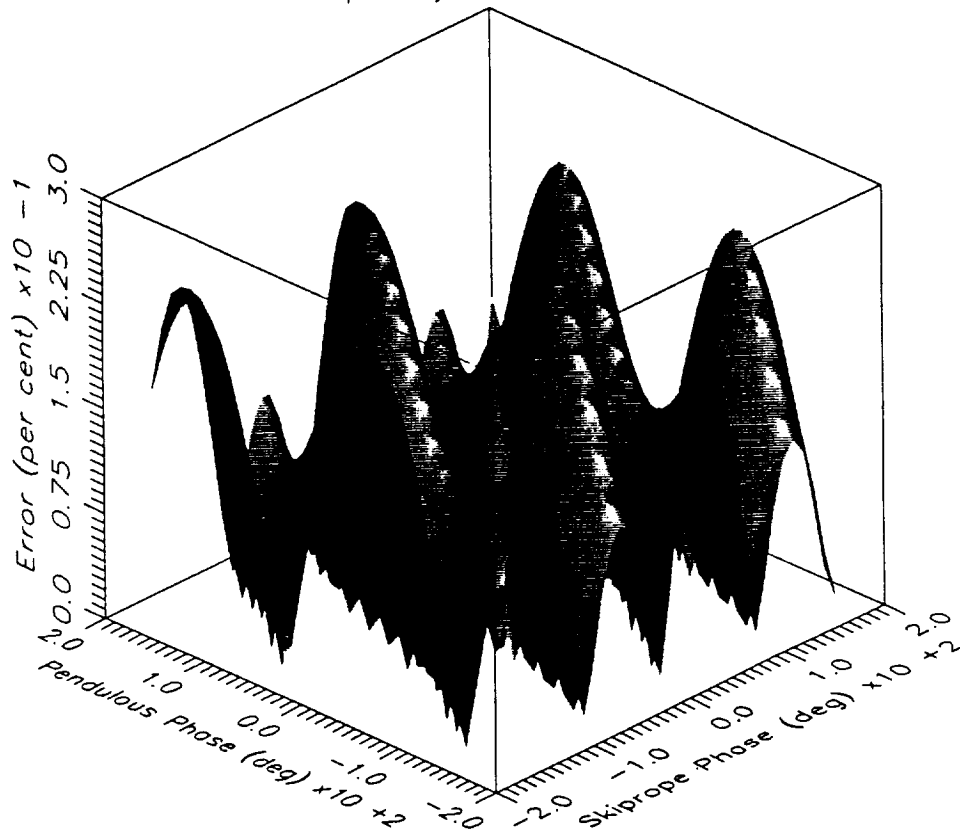
Maximum Amplitude Error = 0.070%
Frequency = 0.0049 Hz



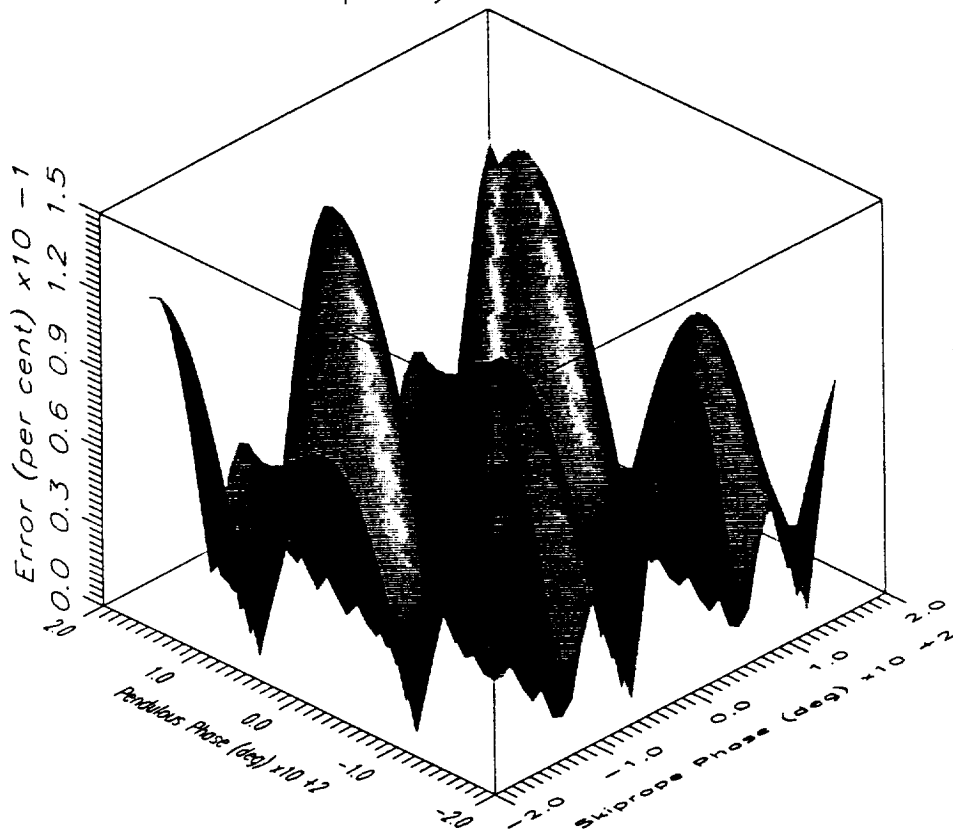
Maximum Amplitude Error = 0.238%
Frequency = 0.0050 Hz



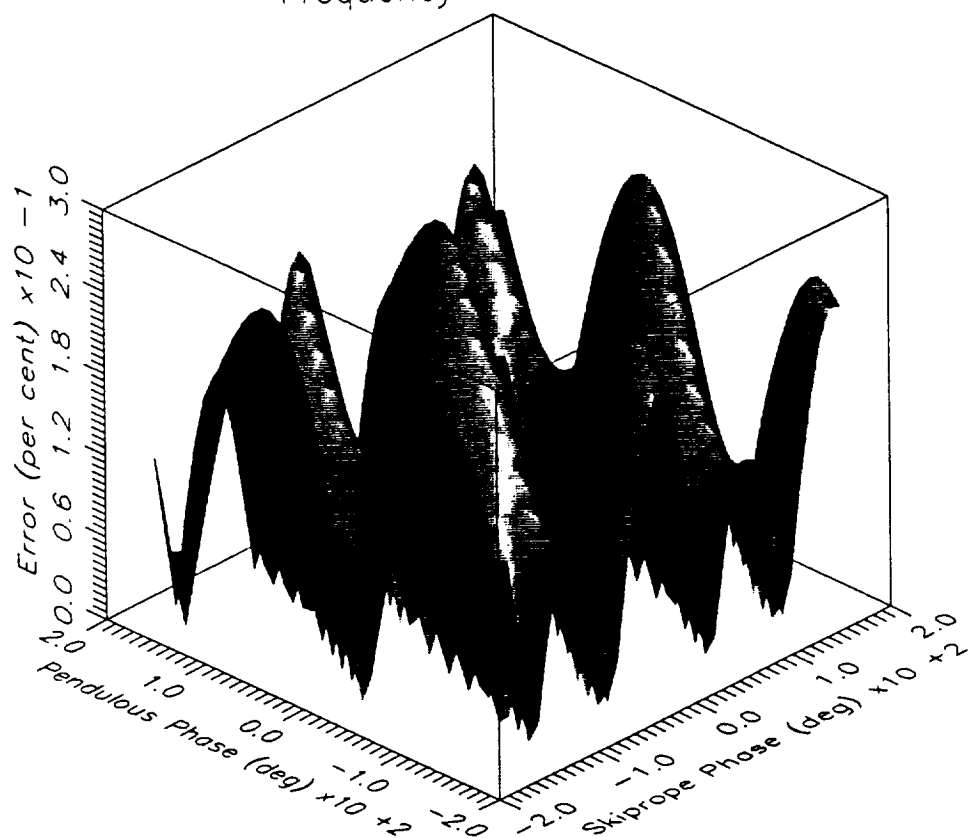
Maximum Amplitude Error = 0.260%
Frequency = 0.0051 Hz



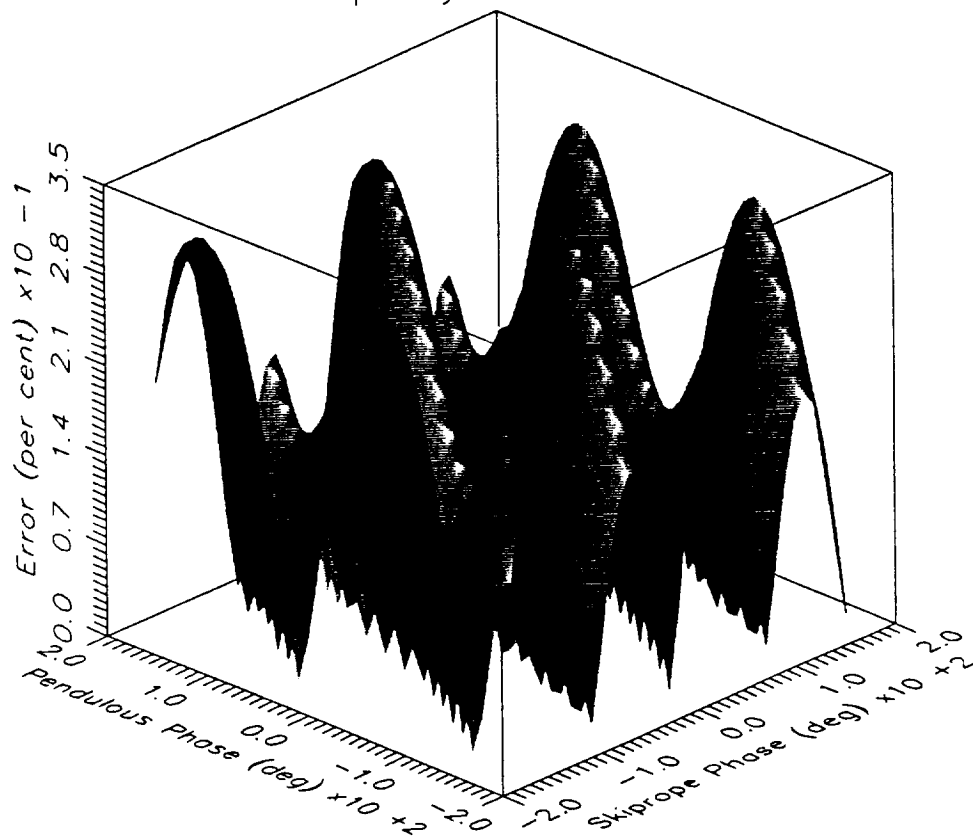
Maximum Amplitude Error = 0.121%
Frequency = 0.0052 Hz



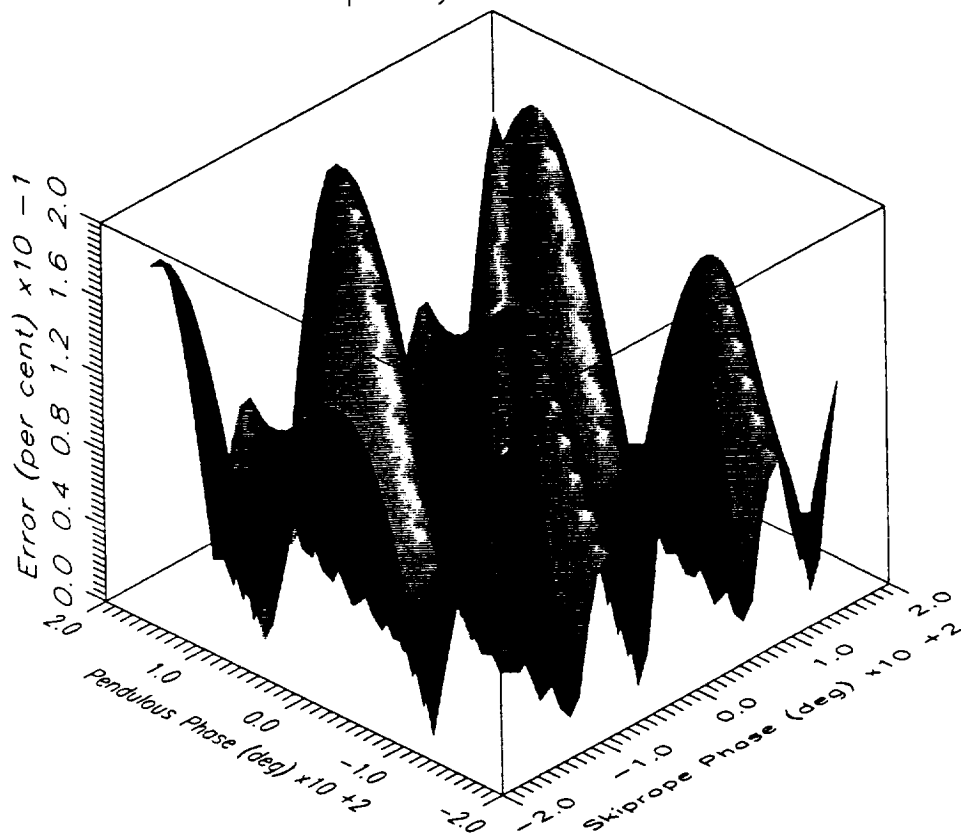
Maximum Amplitude Error = 0.266%
Frequency = 0.0053 Hz



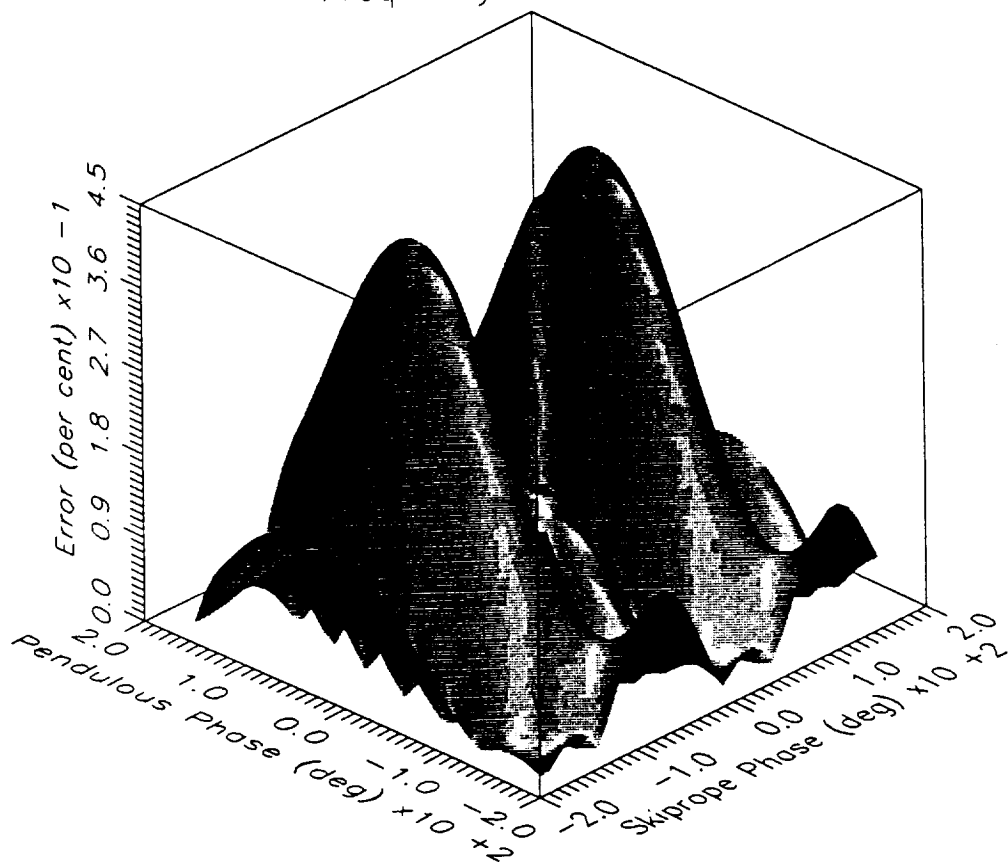
Maximum Amplitude Error = 0.324%
Frequency = 0.0054 Hz



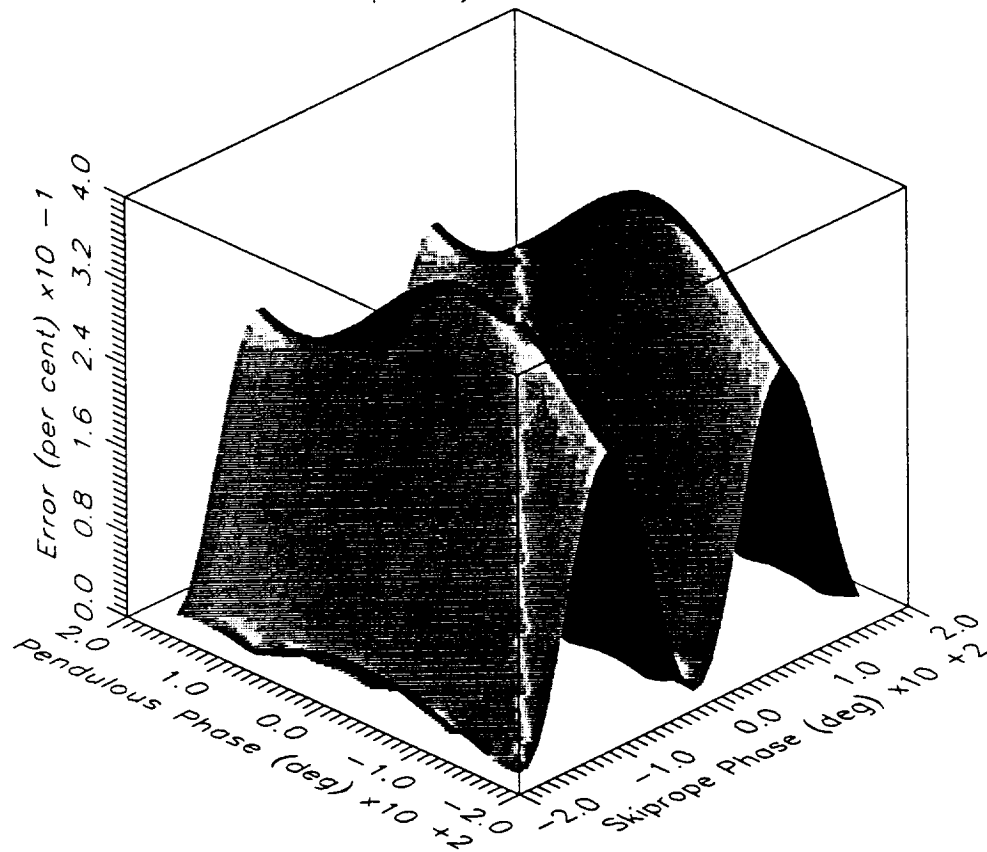
Maximum Amplitude Error = 0.185%
Frequency = 0.0055 Hz



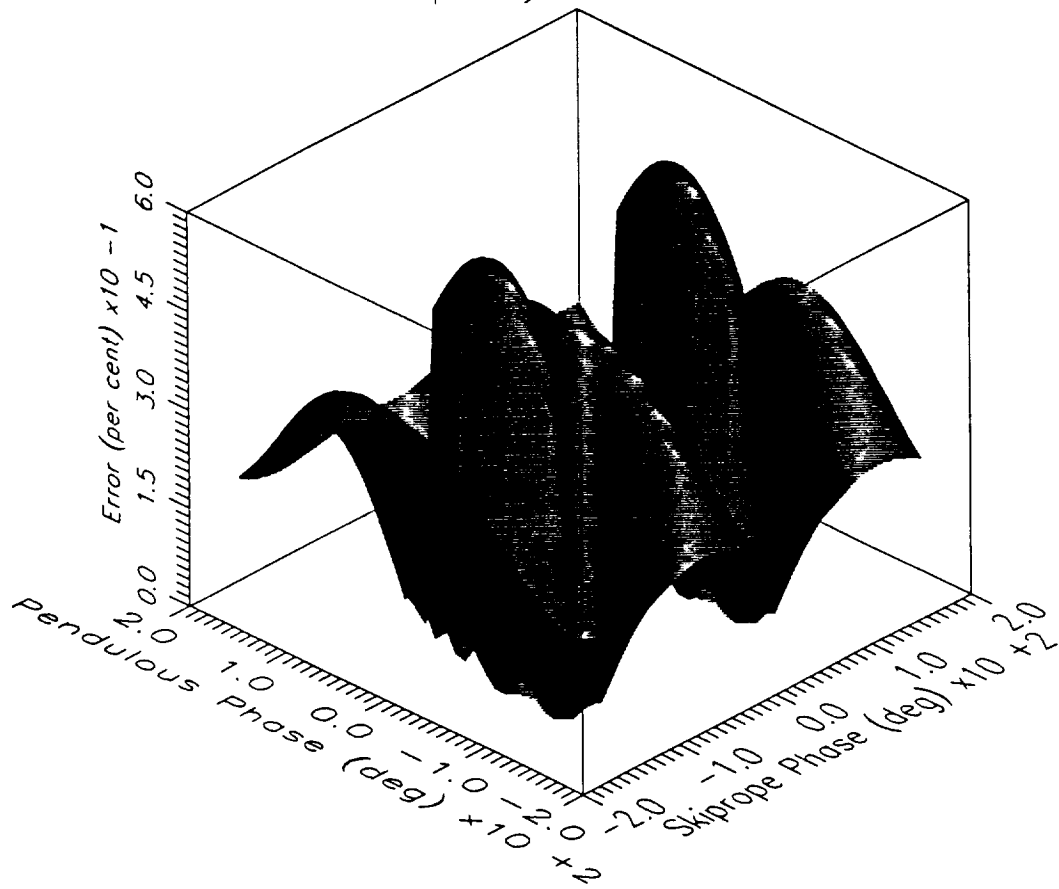
Maximum Frequency Error = 0.434%
Frequency = 0.0045 Hz



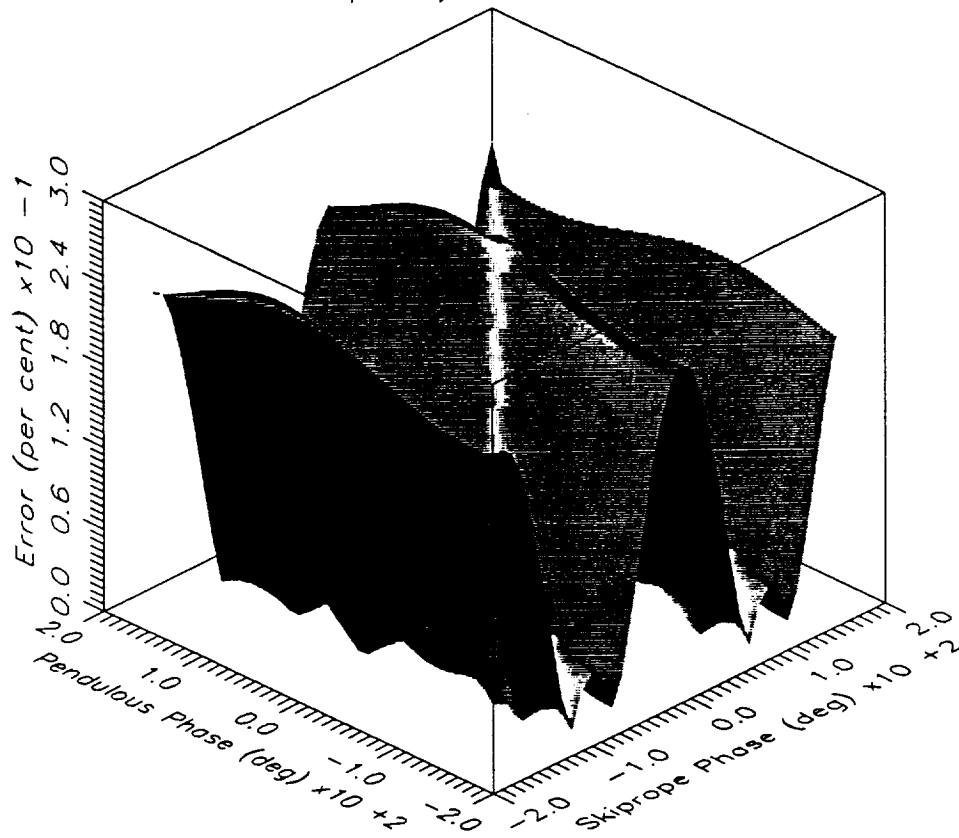
Maximum Frequency Error = 0.370%
Frequency = 0.0046 Hz



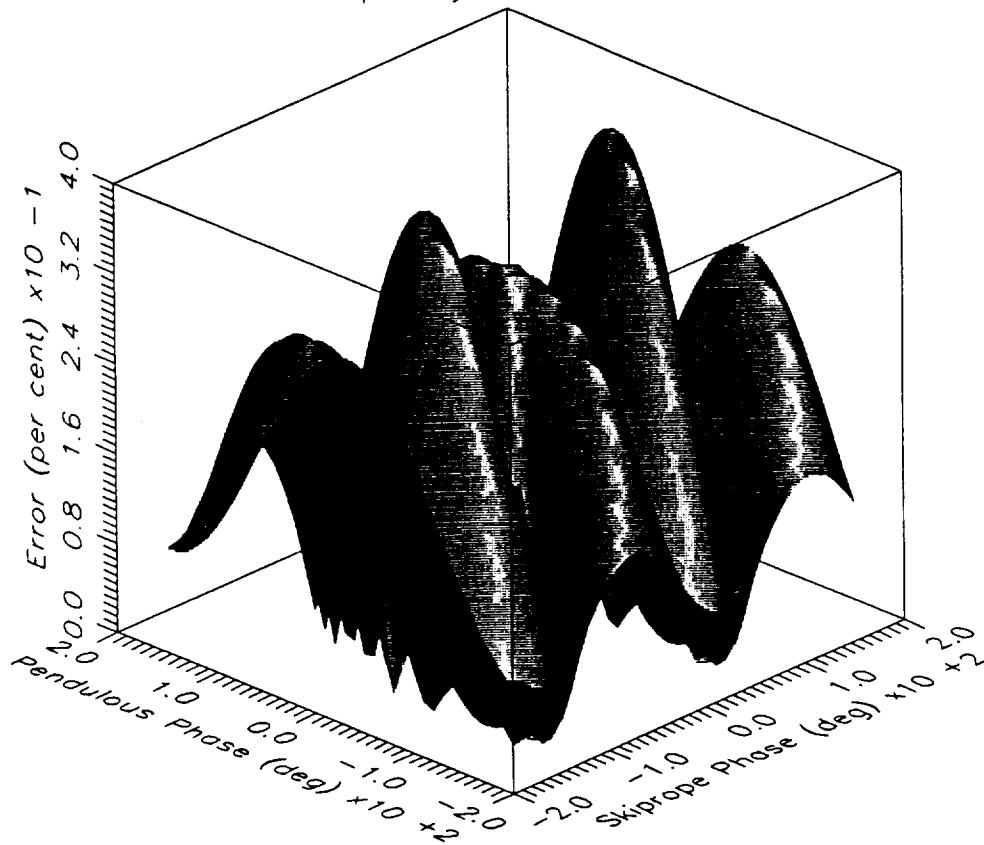
Maximum Frequency Error = 0.590%
Frequency = 0.0047 Hz



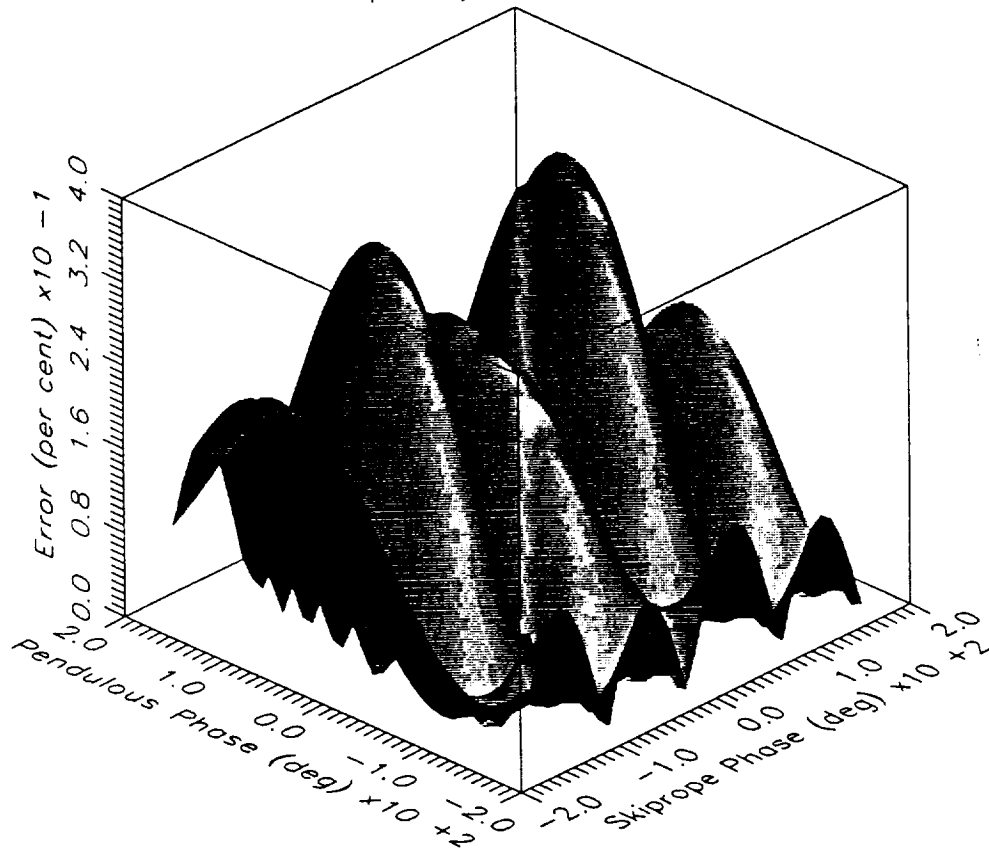
Maximum Frequency Error = 0.267%
Frequency = 0.0048 Hz



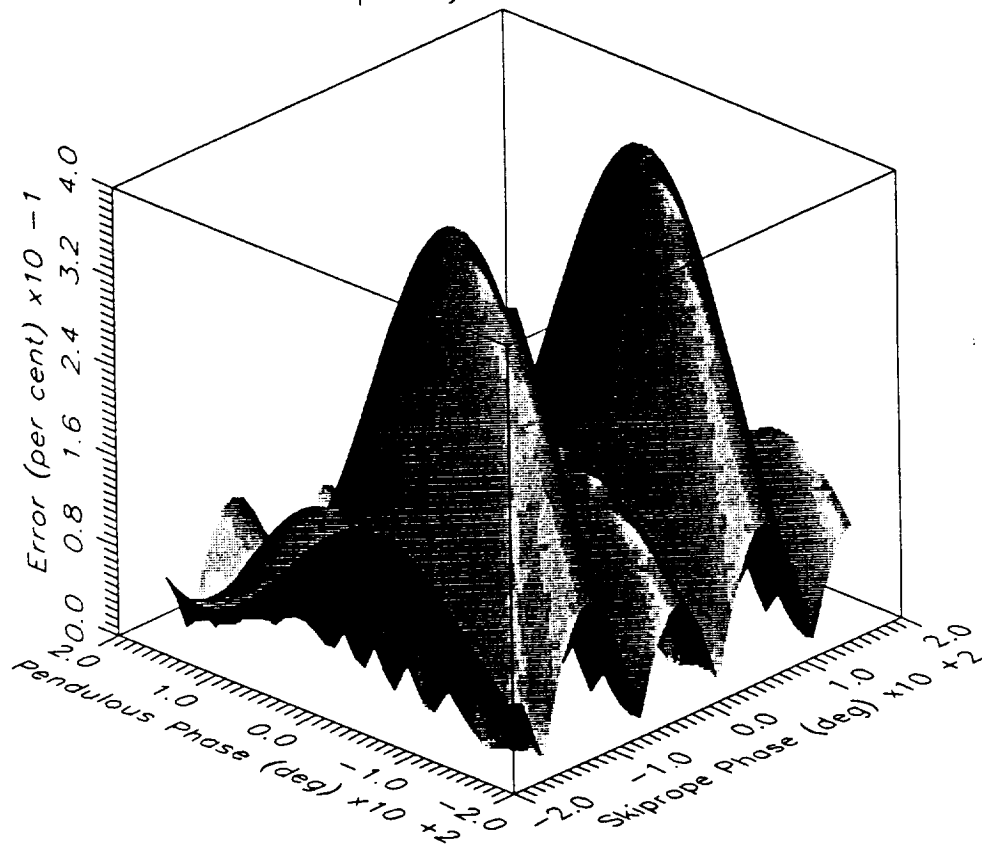
Maximum Frequency Error = 0.408%
Frequency = 0.0049 Hz



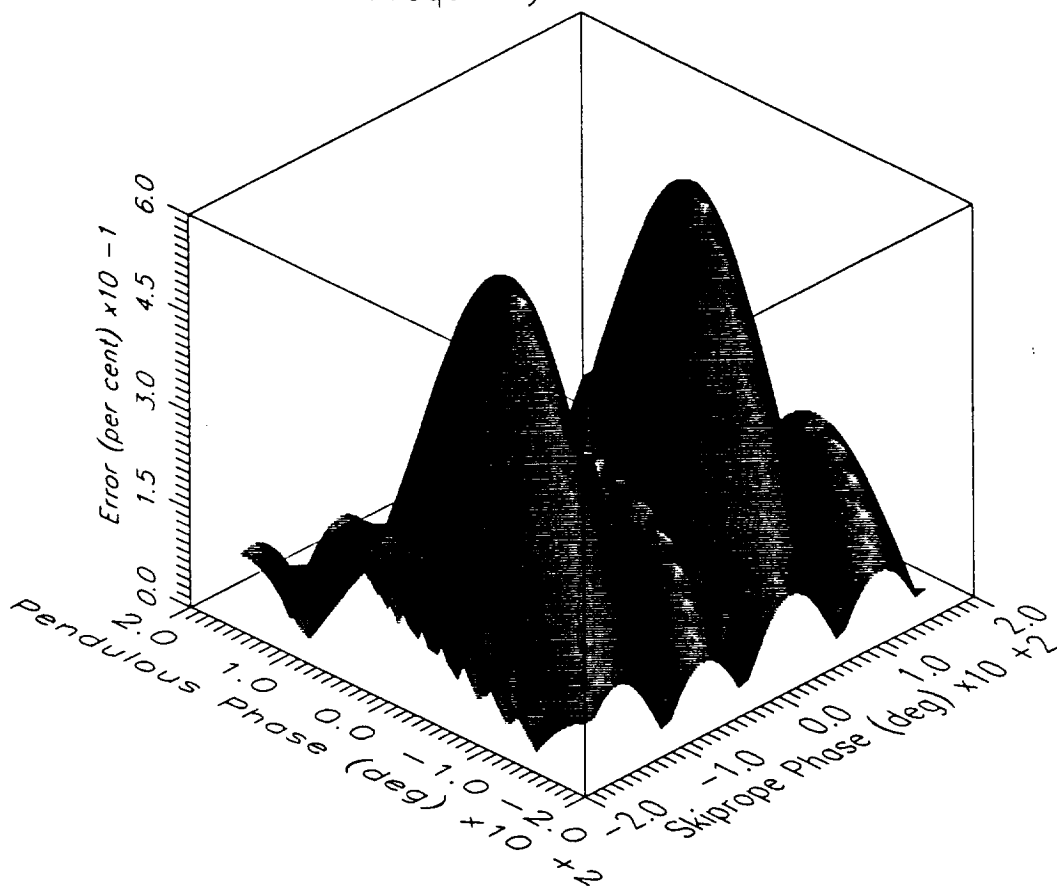
Maximum Frequency Error = 0.369%
Frequency = 0.0050 Hz



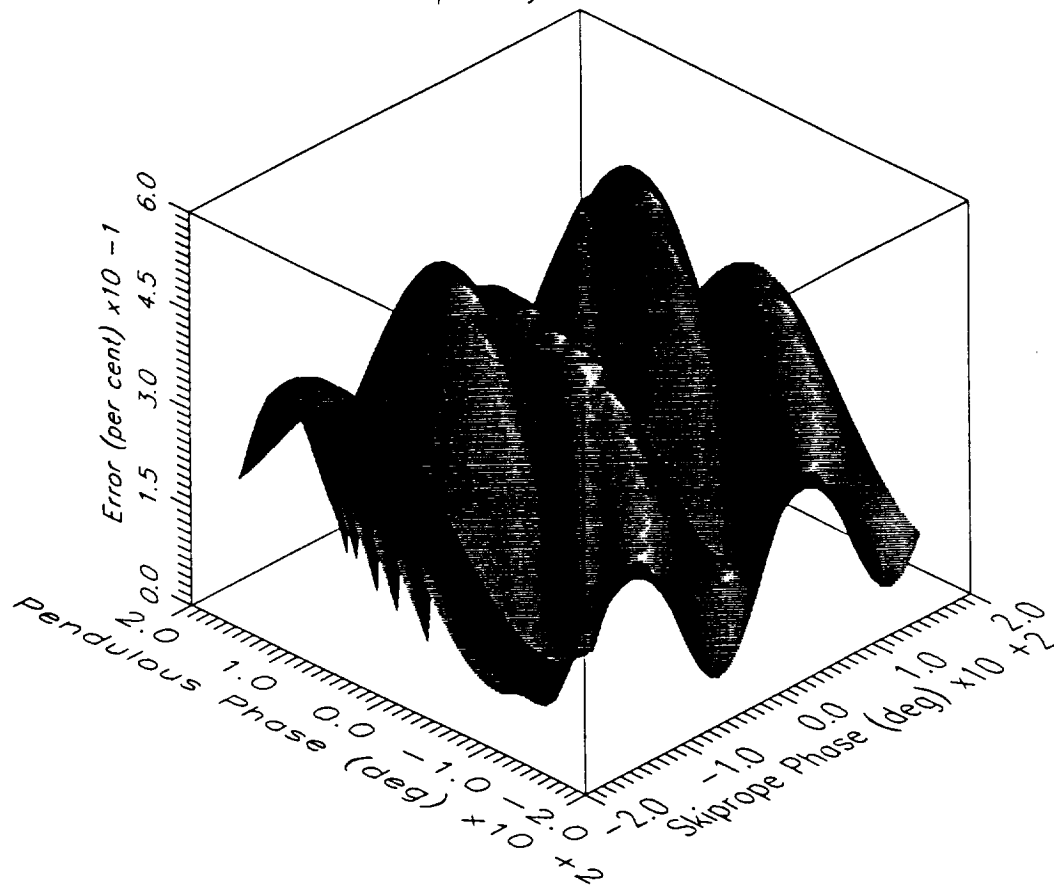
Maximum Frequency Error = 0.405%
Frequency = 0.0051 Hz



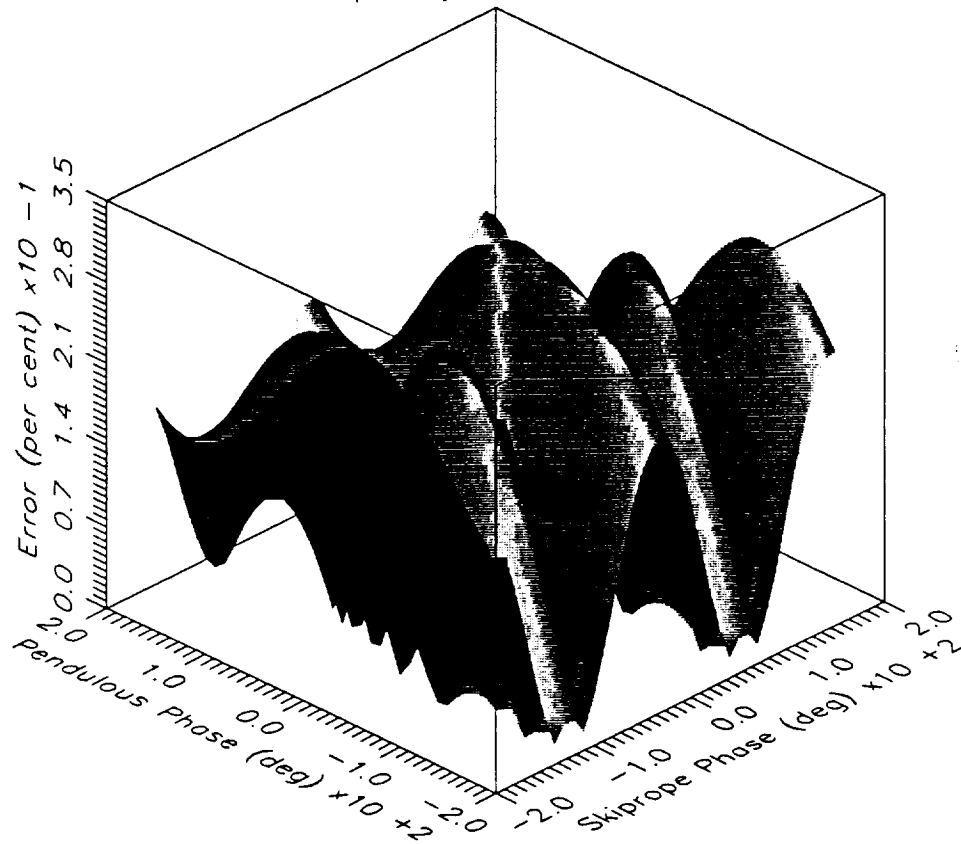
Maximum Frequency Error = 0.578%
Frequency = 0.0052 Hz



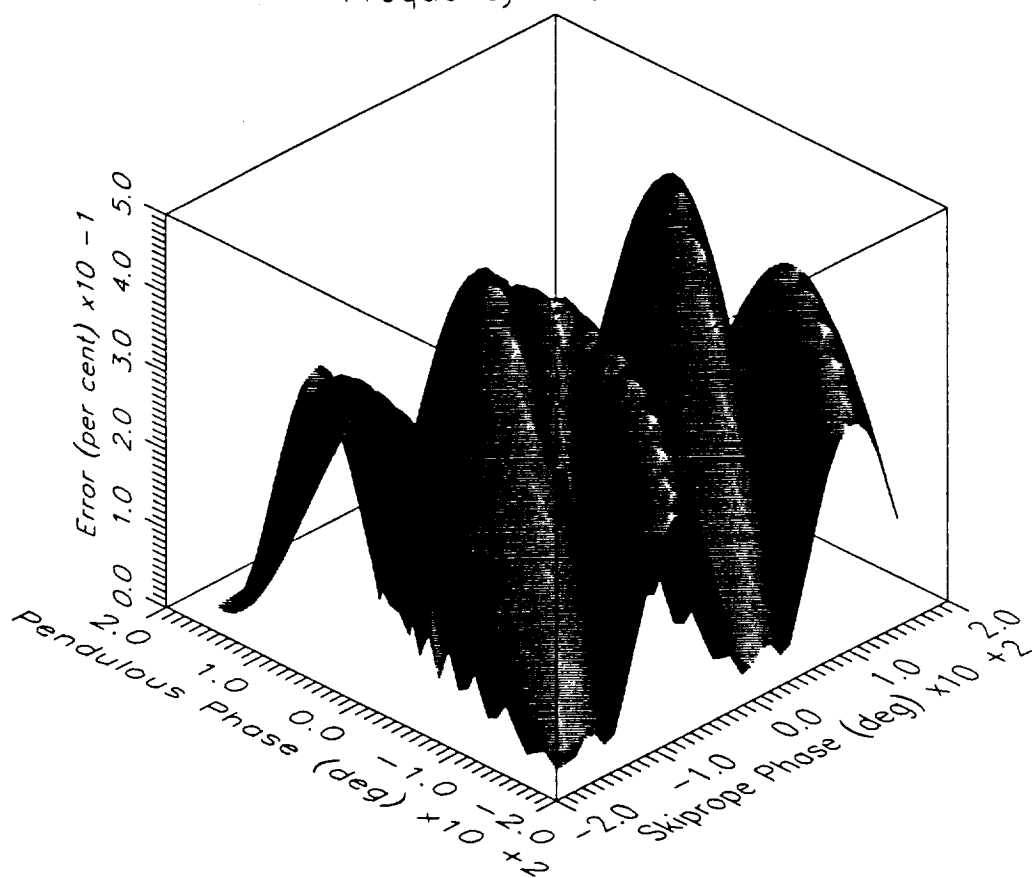
Maximum Frequency Error = 0.550%
Frequency = 0.0053 Hz



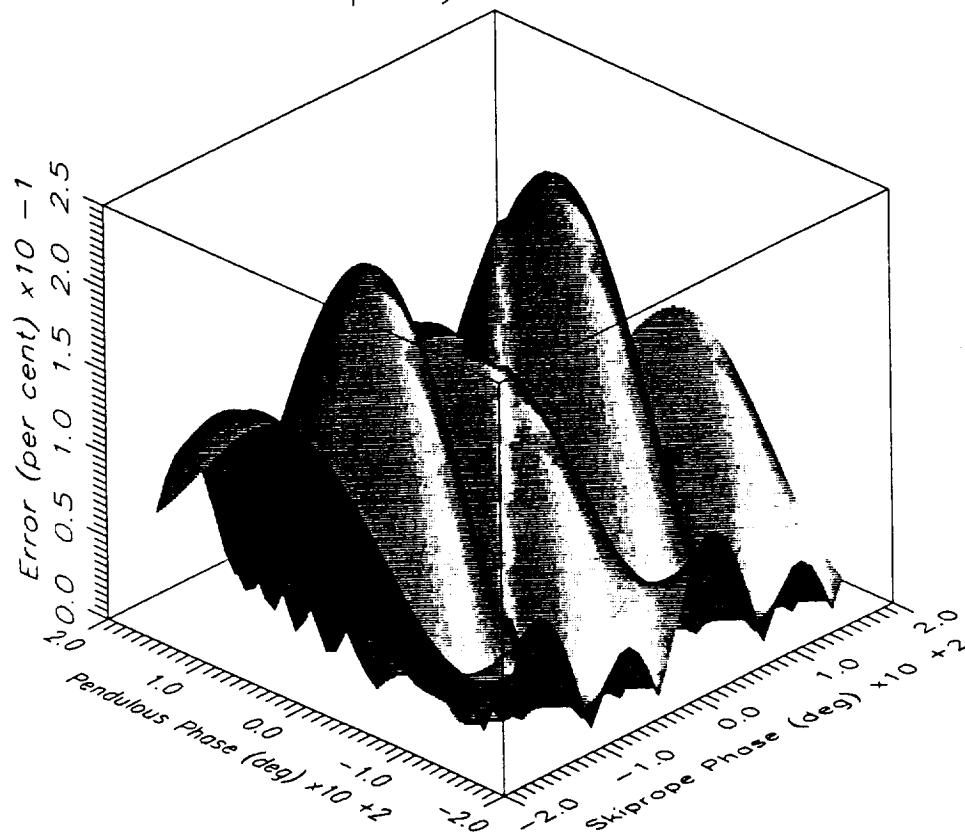
Maximum Frequency Error = 0.308%
Frequency = 0.0054 Hz



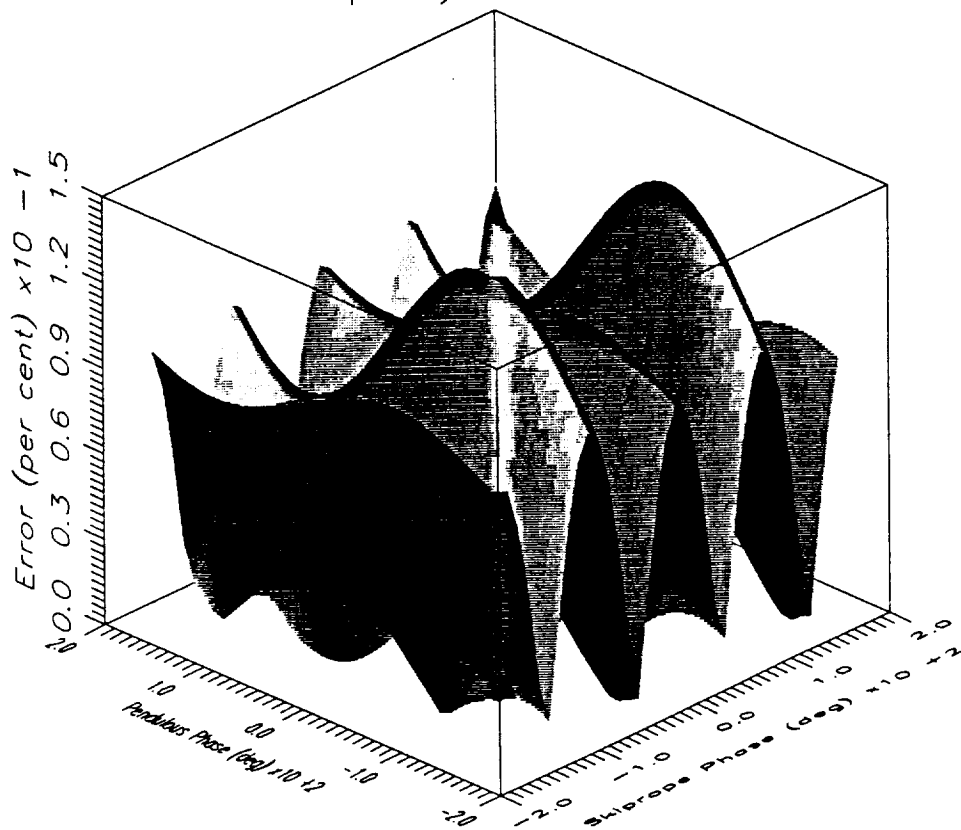
Maximum Frequency Error = 0.497%
Frequency = 0.0055 Hz



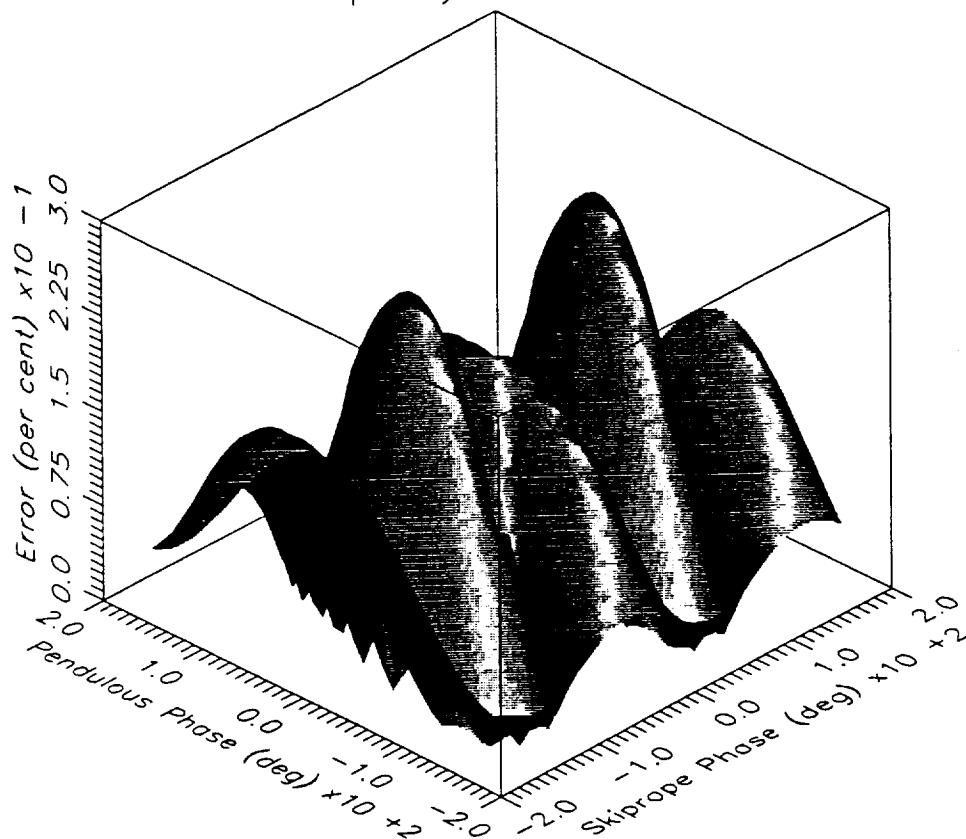
Maximum Phase Error = 0.225%
Frequency = 0.0045 Hz



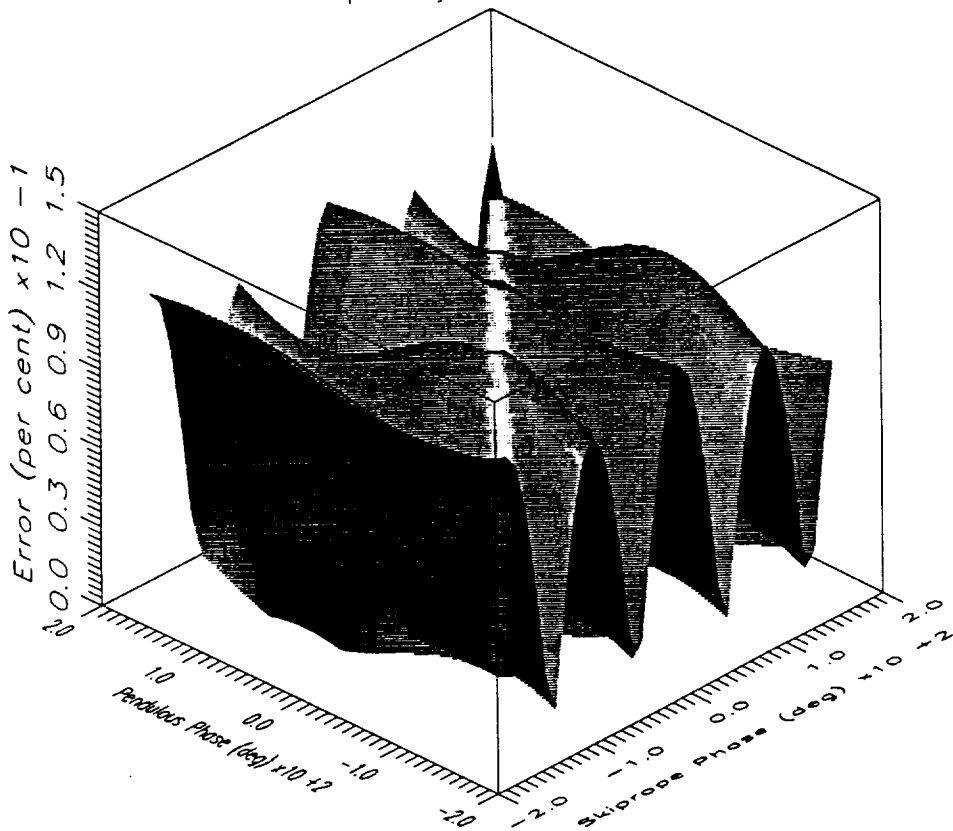
Maximum Phase Error = 0.147%
Frequency = 0.0046 Hz



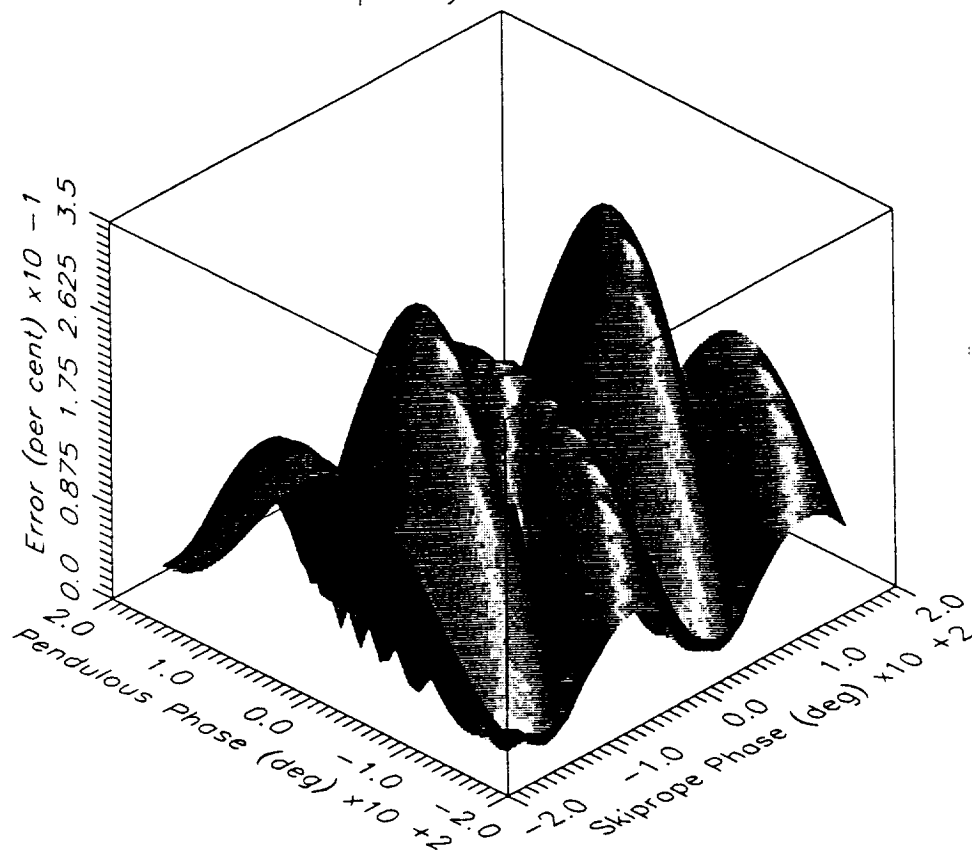
Maximum Phase Error = 0.276%
Frequency = 0.0047 Hz



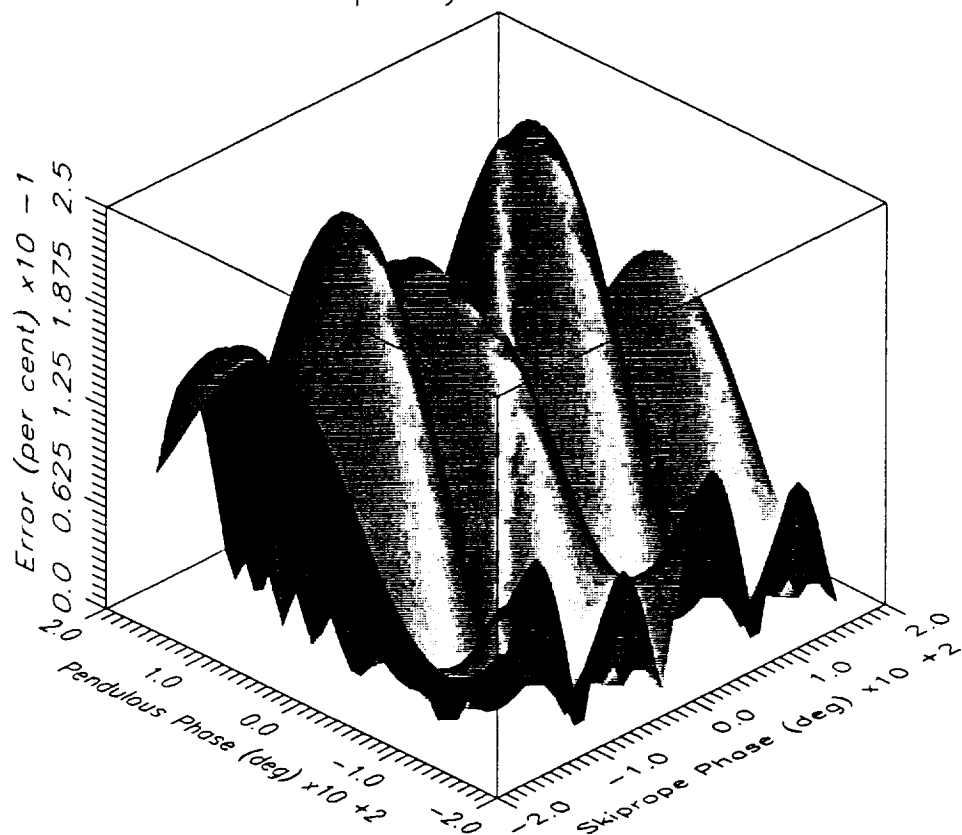
Maximum Phase Error = 0.127%
Frequency = 0.0048 Hz



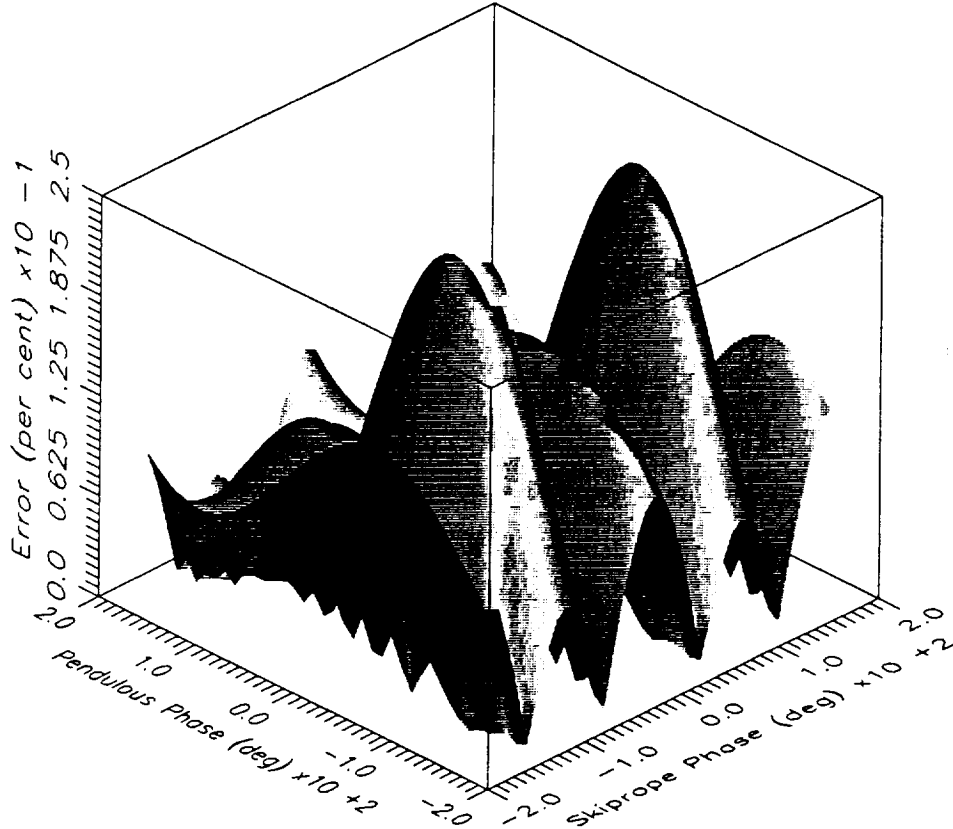
Maximum Phase Error = 0.311%
Frequency = 0.0049 Hz



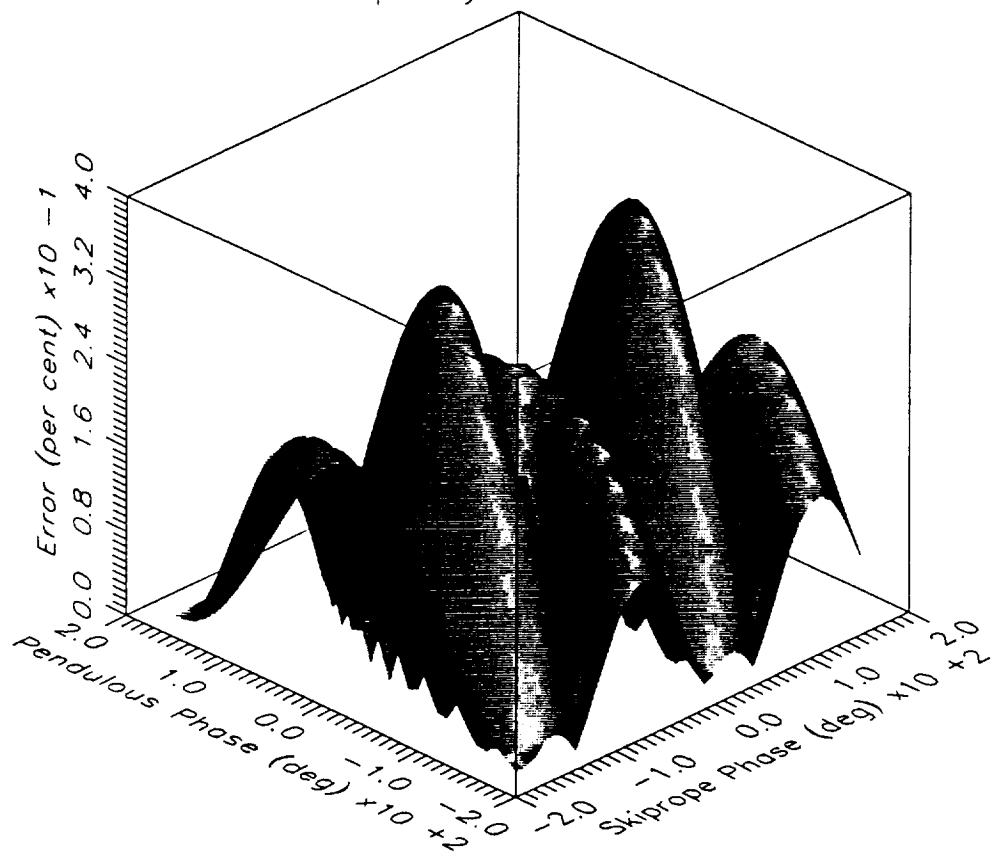
Maximum Phase Error = 0.253%
Frequency = 0.0050 Hz



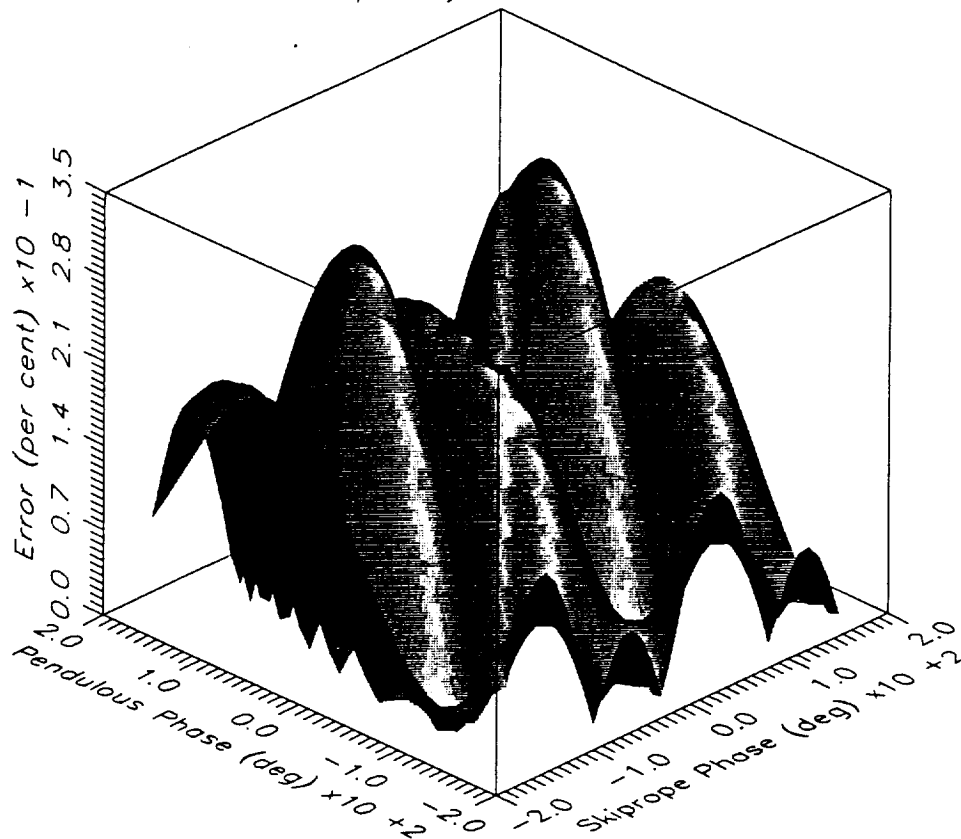
Maximum Phase Error = 0.254%
Frequency = 0.0051 Hz



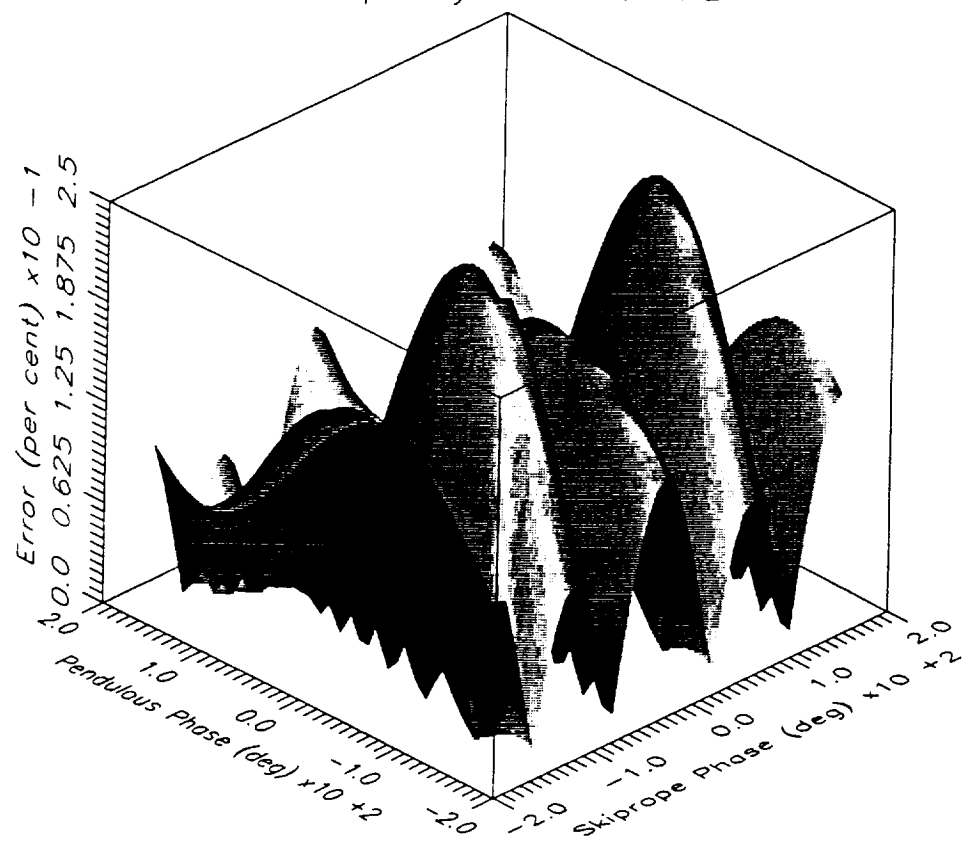
Maximum Phase Error = 0.360%
Frequency = 0.0052 Hz



Maximum Phase Error = 0.320%
Frequency = 0.0053 Hz



Maximum Phase Error = 0.258%
Frequency = 0.0054 Hz



Maximum Phase Error = 0.407%
Frequency = 0.0055 Hz

